

UNIVERZA V LJUBLJANI  
FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Anej Lekše

## **Avtomatizacija delavniškega dnevnika**

DIPLOMSKO DELO

VISOKOŠOLSKI STROKOVNI ŠTUDIJSKI PROGRAM  
PRVE STOPNJE  
RAČUNALNIŠTVO IN INFORMATIKA

MENTOR: dr. Andrej Brodnik

Ljubljana, 2020

COPYRIGHT. Rezultati diplomske naloge so intelektualna lastnina avtorja in Fakultete za računalništvo in informatiko Univerze v Ljubljani. Za objavo in koriščenje rezultatov diplomske naloge je potrebno pisno privoljenje avtorja, Fakultete za računalništvo in informatiko ter mentorja.

*Besedilo je oblikovano z urejevalnikom besedil L<sup>A</sup>T<sub>E</sub>X.*

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogu:

Tematika naloge:

Preveri, ali so glasovni pomočniki v trenutnem stanju primerni za pomoč pri pisanju delavniških dnevnikov. Seznani se z obstoječimi programskimi rešitvami in jih analiziraj. Po analizi področja se loti izdelave svojega sistema za pisanje delavniških dnevnikov, ki vključuje glasovnega pomočnika.



*Na tem mestu zapišite, komu se zahvaljujete za izdelavo diplomske naloge. Pazite, da ne boste koga pozabili. Utegnil vam bo zameriti. Temu se da izogniti tako, da celotno zahvalo izpustite.*



Kavi. Hvala.



# Kazalo

## Povzetek

## Abstract

<b>1</b>	<b>Uvod</b>	<b>1</b>
1.1	Opis domene raziskave . . . . .	1
1.2	Struktura diplomske naloge . . . . .	2
<b>2</b>	<b>Pregled problema in rešitve</b>	<b>3</b>
2.1	Problem . . . . .	3
2.2	Obstoječe rešitve . . . . .	5
<b>3</b>	<b>Načrtovanje in razvoj sistema za pisanje opisov tehnoloških postopkov</b>	<b>11</b>
3.1	Definicija funkcionalnosti . . . . .	11
3.2	Načrt sistema za sestavljanje opisov tehnoloških postopkov . .	14
3.3	Uporabljene tehnologije . . . . .	16
3.4	Strežnik . . . . .	19
3.5	Implementacija Alexa „Skill-a“ . . . . .	39
3.6	Implementacija mobilne aplikacije . . . . .	46
3.7	Testiranje sistema OpenReport . . . . .	58
3.8	Ovrednotenje funkcionalnosti . . . . .	58
<b>4</b>	<b>Zaključek</b>	<b>61</b>





# Seznam uporabljenih kratic

kratica	angleško	slovensko
<b>API</b>	Application Programming Interface	vmesnik za programiranje aplikacije
<b>AWS</b>	Amazon Web Services	Amazonove spletne storitve
<b>FIFO</b>	First In First Out	Prvi not, prvi ven
<b>HTML</b>	HyperText Markup Language	Označitveni jezik za hipertekst
<b>HTTP</b>	HyperText Transfer Protocol	Protokol za prenos hiperteksta
<b>HTTPS</b>	HyperText Transfer Protocol Secure	Protokol za prenos hiperteksta, varna različica?
<b>IP</b>	Internet Protocol	Internetni protokol
<b>JSON</b>	JavaScript Object Notation	JavaScript zapis objektov
<b>JWT</b>	JSON Web Token	JSON spletni žeton
<b>LIMS</b>	Laboratory Information Management System	laboratorijski sistem za uporabljjanje informacij
<b>MIT</b>	Massachusetts Institute of Technology	Tehnološki inštitut Massachusettsa
<b>MVVM</b>	Model View View-Model	Model Pogled Pogled-Model
<b>NLU</b>	Natural Language Understanding	Razumevanje naravnega govora
<b>SQS</b>	Simple Queue Service	preprosta vrstna storitev
<b>UI</b>	User Interface	uporabniški vmesnik
<b>URL</b>	Universal Resource Locator	Univerzalni lokator virov
<b>VUI</b>	Voice User Interface	glasovni uporabniški vmesnik
<b>XAML</b>	Extensible Application Markup Language	razširljiv aplikacijski označitveni? jezik

# Povzetek

**Naslov:** Avtomatizacija delavníškega dnevnika

**Avtor:** Anej Lekše

Diplomsko delo obravnava področje vpeljave komercialno dostopnih glasovnih pomočnikov v programske rešitve kot sredstvo za pohitritev uporabe. V nalogi raziščemo, kako v računalniški sistem vpeljemo glasovnega pomočnika in vpliv glasovnega pomočnika na uporabo sistema. Cilj je implementirati in preizkusiti računalniški sistem z glasovnim pomočnikom, s pomočjo katerega lahko narekujemo zapiske med delom. Te zapiske pa lahko naknadno urejamo s pomočjo mobilne aplikacije. Uporabili smo Amazon Alexa zaradi enostavne izdelave lastnih programov (t.i. „Skill-ov“). Ugotovili smo, da Amazon Alexa ni bila zmožna razpozнатi izgovorjenih stavkov z veliko variirajočimi besedami. Uspešna je bila pri razpoznavanju stavkov, ki so se v večini ali v celoti ujemali z naučenim razpoznavnim modelom. Kljub temu je čas, ki je potreben, da Alexa razpozna prejet glasovni ukaz prepočasen, da bi bil sistem učinkovit.

**Ključne besede:** mobilni razvoj, glasovni pomočniki, razpoznava glasu, informacijski sistemi.



# Abstract

**Title:** Workshop report automatization

**Author:** Anej Lekše

This thesis deals with the process of incorporating a commercially available voice assistant into a software solution as a means of optimizing the workflow. The goal of thesis was implementing and testing a system with a voice assistant that could be used to write workshop report steps during work. These notes can then be edited with a mobile application. We have used Amazon Alexa, as it offers simple programming with Alexa Skills. We discovered that Amazon Alexa was unsuitable for literal dictation of longer, largely varying text. It was successful at recognising phrases that fully or mostly matched its learned model. Regardless, Alexa's ability to process custom voice commands, at the time of this research, was too slow to be deemed efficient.

**Keywords:** mobile development, voice assistants, voice recognition, information systems.



# Poglavlje 1

## Uvod

### 1.1 Opis domene raziskave

Diplomsko delo obravnava področje vpeljave glasovnih pomočnikov v programske rešitve kot sredstvo za izboljšanje učinkovitosti dela.

Za področje, ki ga želimo izboljšati, smo si izbrali pisanje delavniškega dnevnika ali opisa tehnološkega postopka. Kljub temu se bo naš sistem, z manjšimi spremembami, lahko uporabil tudi za pisanje drugačnih vrst poročil in navodil. Med te spadajo na primer kuharski recepti, laboratorijski dnevniki, itd.

Opis tehnološkega postopka ali delavniški dnevnik je dokument, ki po korakih predstavi postopek izdelave izdelka. Delavniški dnevniki imajo lahko definirane tudi kontrolne postopke za izdelek, orodje, ki ga rabimo za izdelavo, in seznam možnih nevarnosti pri delu.

Delavniški dnevnik je sestavljen iz zaporedja korakov, ki si sledijo v časovnem zaporedju. Vsak korak sestavlja opis postopka in predvideno trajanje. Koraki lahko vsebujejo tudi slikovne razlage.

Opisi tehnološkega postopka se najpogosteje uporabljajo v proizvodnih obratih tovarn kot navodila in oporne točke za sestavljalce.

V sklopu diplomske naloge smo želeli izdelati specializiran sistem za pisanje delavniških dnevnikov, nato pa preveriti, ali lahko z vpeljavo glasovnega

pomočnika delo s tem sistemom pohitrimo. Ta sistem bi sestavljal glasovni pomočnik, ki bi služil za narekovanje opomb, mobilna aplikacija, preko katere bi lahko urejali zapiske in strežnik, ki bi hranił podatke.

Raziskati želimo predpostavko, da lahko z učinkovito uporabo glasovnega pomočnika zmanjšamo čas interakcije z mobilno aplikacijo.

## 1.2 Struktura diplomske naloge

Diplomsko delo pričenjam s predstavitvijo področja raziskave in kratko opišemo problem in možno rešitev. Opredelimo problematiko naše raziskave.

Začnemo z raziskavo obstoječih rešitev za pisanje opisa tehnološkega postopka. Nato opišemo, kaj trenutne rešitve te problematike ponujajo in njihove prednosti ter slabosti.

V naslednjem poglavju se lotimo načrtovanja specializiranega sistema z glasovnim pomočnikom za pomoč pri pisanju delavnih dnevnikov. Nato se lotimo opisa tehnologij, ki smo jih pri pisanju diplome uporabili. Natančno definiramo funkcionalnosti sistema, utemeljimo odločitev za izbiro Amazon Alexe, AWS SQS, AWS Lambda in ogrodja Xamarin.

Nato predstavimo Alexa „Skill“, strežniški program in mobilno aplikacijo. Funkcionalnosti sistema testiramo in analiziramo.

V predzadnjem poglavju opišemo možnosti nadaljnega razvoja projekta.

# Poglavlje 2

## Pregled problema in rešitve

### 2.1 Problem

#### 2.1.1 Kaj vsebuje delavniški dnevnik?

Delavniški dnevnik je splošen izraz za tehniški dokument, ki po korakih predstavi potek tehnološkega postopka [5]. V podjetju Hisense Gorenje Europe takšne dokumente imenujejo opis tehnološkega postopka (slika 2.1).

Delavniški dnevnik sestavlja zaporedje korakov, ki si sledijo v časovnem vrstnem redu. Vsak korak delavniškega dnevnika sestavlja opis postopka posamezne naloge v procesu in njeno predvideno trajanje. Delavniški dnevnički lahko vsebujejo tudi slikovne razlage.

Delavniški dnevniki imajo lahko definirane tudi kontrolne postopke za izdelek, orodje, ki ga potrebujemo za izdelavo in seznam možnih nevarnosti pri delu.

Delavniški dnevniki se uporabljam v proizvodnih obratih tovarn kot navodila in oporne točke za sestavljalce. Pogosto se uporabljam tudi kot učni pomoček za spremeljanje napredka npr. dijakov vajencev na praksi [5, 10].

C	D	E	F	G	H	I	J	K
Delovni	Postopek	Kratki tekst postop.	Osnovna ko	Stand.	Enota stand.	Stand.	Stand.	Stevilo zaposlenih
OV847P 0010		Operacija planiranja	1	1 S	41.9	0		0
OV847A 0020		sestavljalec2- trojček	1	0 S	47	3		3
OV660B 0030		kontrola ohišij	1	0 S	47	1		1
OV660A 0040		čiščenje po puru	1	0 S	47	1		3
OV847A 0050		sestavljalec 2	1	0 S	45	15		15
OV847A 0060		sestavljalec 1	1	0 S	45	30		30
OV847B 0070		strojniki embalirnega stroja	1	0 S	47	1		1
OV847B 0080		pregledovalec sklopov	1	0 S	47	6		6
OV847O 0090		mehanik gospodinjskih aparatov	1	0 S	47	3		3
OV847B 0100		varilec gasflux, lak popravilo (1,5)	1	0 S	47	3		2
OV847S 0110		operacija brez delavca	1	27 S	47	0		0
OV847S 0120		vakuum krog	1	22 S	47	0		0
OV847S 0130		100% kontrola	1	19 S	47	0		0
OV847S 0140		povezava montaža - opremljanje	1	6 S	47	0		0
OV847A 0150		Vijačenje nosilca predala	1	0 S	47	1		1
OV847A 0160		Vstavljanje ventilatorja	1	0 S	47	1		1
OV847A 0170		Sestava profilov	1	0 S	47	1		1
OV847A 0180		Odstranjevanje folije	1	0 S	47	2		2
OV847P 0190		Sestava steklene police	1	0 S	19.7	1		4.1
OV847A 0200		Feniranje bulžirke	1	0 S	22.32	1		1
OV784P 0010		Linija za navijanje uparjal - planira	1	1 S	140	0		0
OV784S 0020		navijanje	1	1 S	140	0		0
OV784O 0030		Pripraviti celico, naviti cev	1	0 S	140	1		1
OV784A 0040		Spojiti celico in CPU	1	0 S	136	1		1
OV784A 0050		He detekcija	1	0 S	140	1		1

Slika 2.1: V tabelo izvožen opis tehnološkega postopka za izdelavo hladilnika v podjetju Hisense Gorenje Europe

Delavniški dnevnik se lahko sestavlja med delom ali po opravljenem delu.

### 2.1.2 Definicija problematike

V tej diplomski nalogi raziščemo možne metode izdelave delavniškega dnevnika oz. opisa tehnološkega postopka. Raziskati želimo prednosti in slabosti trenutnih metod sestavljanja teh dokumentov. V naslednji fazi raziskave želimo izdelati specializiran sistem za sestavljanje opisov tehnoloških postopkov. Raziskati želimo vpeljavo glasovnega pomočnika v sistem za sestavljanje delavniških dnevnikov in kako uporaba glasovnega pomočnika vpliva na uporabo takega sistema.

Izhajajoč iz navedenega opredeljujemo problematiko diplomskega dela: zasnuj in implementiraj sistem za sestavljanje delavniških dnevnikov, vanj vpelji glasovnega pomočnika in sistem preizkusi.

## 2.2 Obstojče rešitve

### 2.2.1 Pisanje na papir

Najstarejša metoda za izdelavo takšnega dokumenta je zapis na list papirja (slika 2.2).

Opis tehnološkega postopka		
Naslov:	ČIŠČENJE TIPKOVNICE	
Opis:	ČIŠČENJE MEHANSKE TIPKOVNICE	
Nevarnosti:	/	
Korak	Opombe	Trajanje
PRIPRAVI DELOVNO MIZO		5 MIN
PRIPRAVI TIPKOVNICO		1 MIN
ODSTRANI POKROVE TIPK	ORODJE JE PRILOŽENO V ŠKATLI	10 MIN
POKROVE TIPK NAMOČI V MILNICO	PUSTI VSAJ 10 MIN	minimalno 10 MIN
OBRISI POKROVE TIPK DO POVSEM SUHEGA		30 MIN
DAJ POKROVE TIPK NAZAJ NATPK.		8 MIN

Slika 2.2: Opis tehnološkega postopka za čiščenje tipkovnice na papirju.

Prednosti uporabe papirja in pisala so, da pri delu izdelovalec ne potrebuje računalnika in cenovna ugodnost. Slabosti takšnega postopka so:

- omejitve glede velikosti prostora, namenjenega vsakemu koraku,
- problematično dopisovanje in urejanje obstoječih korakov,
- zahtevnejše arhiviranje od računalniških datotek,

- občutljivost papirja na fizične poškodbe (trganje, mečkanje, vnetljivost,...).

### 2.2.2 Pisarniški programi

Opis tehnološkega postopka lahko izdelamo v pisarniških programih, na primer Microsoft Word ali LibreOffice Writer.

Ta pristop reši nekatere slabosti uporabe papirja in pisala za pisanje opisa tehnološkega postopka. Korake lahko enostavno dodajamo in urejamo. Prav tako je možno dodajati slikovno gradivo. Pisarniški programi omogočajo izpis dokumenta na tiskalnik, če želimo imeti dokument natisnjen na papirju.

Kljud temu uporaba te metode prinese nove slabosti:

- če imamo dokument shranjen na več mestih, moramo ob spremembah zagotoviti, da se spremenijo vse shranjene različice.
- Slikovno gradivo je vezano na dokument. Ob spremembah moramo spremeniti celoten dokument, ne le slike.
- Možnost izgube ali izbrisa podatkov.

V sklopu diplomske naloge smo napisali preprost opis tehnološkega postopka (slika 2.3) s programom LibreOffice Writer [9]. Dokument brez vsebine se lahko pri naslednjem opisu uporabi kot obrazec.

Opis tehnološkega postopka

Avtor: Anej Lekše

Datum: 12. 8. 2020

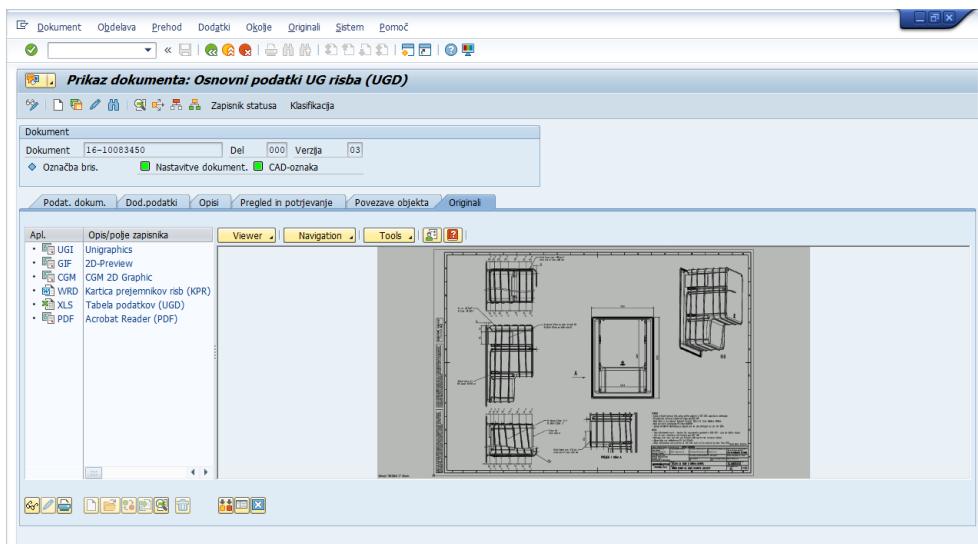
<b>Naslov:</b>	<b>Čiščenje tipkovnice</b>
<b>Opis:</b>	Čiščenje mehanske tipkovnice
<b>Nevarnosti:</b>	Ni nevarnosti

Korak	Opombe	Trajanje
Pripravi delovno mizo in tipkovnico		5 min
Odstrani pokrove tipk	Uporabi orodje, ki je bilo priloženo ob tipkovnici	10 min
Pokrove tipk namoči v milnico	Pusti v vodi vsaj 10 minut	10 min
Obriši pokrove tipk do suhega		30 min
Vstavi pokrove tipk na svoja mesta		8 min

Slika 2.3: Preprost opis tehnološkega postopka za čiščenje tipkovnice, napisan v pisarniškem programu.

### 2.2.3 Specializirani moduli za poslovne informacijske sisteme

Podjetja kot so npr. Hisense Gorenje Europe, za sestavljanje delavninskih dnevnikov uporabljajo specializirane module za lastne informacijske sisteme. Primer je prikazan modul za informacijski sistem SAP (na sliki 2.4).



Slika 2.4: Prikazan korak opisa tehnološkega postopka s programom SAP.

Do tega opisa tehnološkega postopka se lahko dostopa iz računalnikov na delovnih mestih, kjer se izvajajo koraki, opisani v dokumentu. Opis tehnološkega postopka v sistemu SAP sestavlja:

- podatki o izdelku,
- dodatne opombe,
- opisi korakov,
- definicija kontrolnih postopkov in pregleda,
- CAD izris izdelka.

Specializiran informacijski sistem, kot je SAP, je tesno povezan s proizvodno linijo in prilagodljiv za potrebe proizvodnega obrata, ki ga uporablja. Pri sistemu SAP se podatki primarno hranijo na osrednjem strežniku, varnostna kopija pa se inkrementalno dela na geografsko ločen rezervni strežnik. Glavne slabosti takšnega sistema so:

- potrebna proizvodna infrastruktura, ki jo sistem rabi za optimalen izkoristek in

- cena, ki je potrebna za implementacijo takega sistema v proizvodno linijo.



# Poglavlje 3

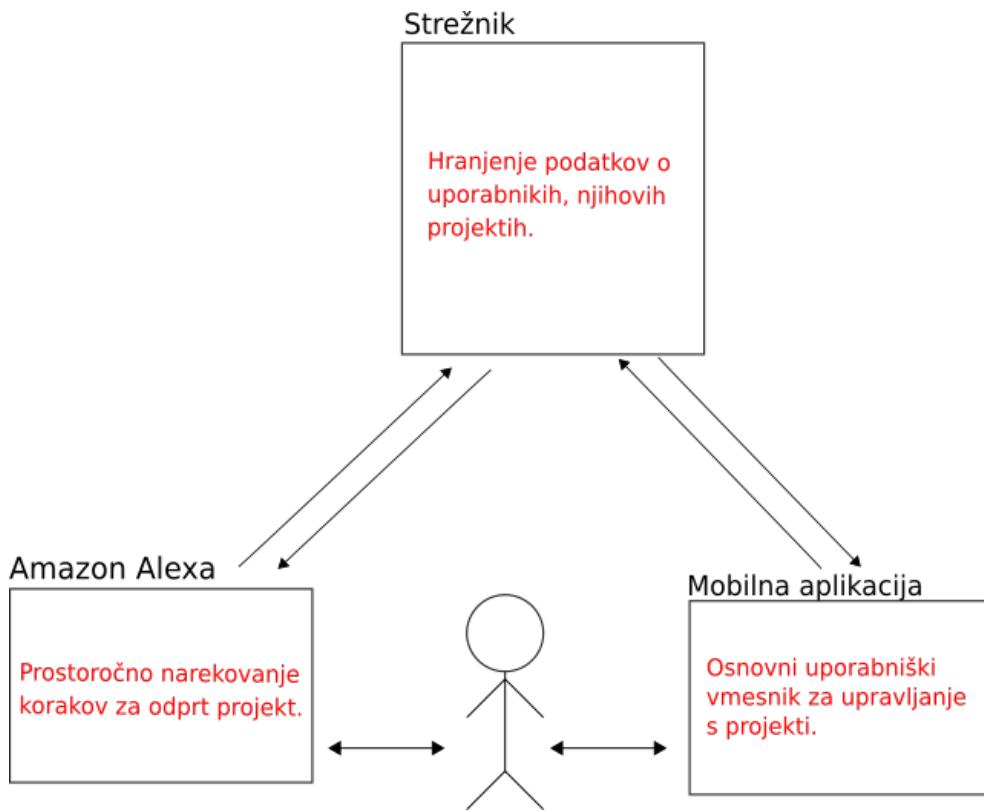
## Načrtovanje in razvoj sistema za pisanje opisov tehnoloških postopkov

### 3.1 Definicija funkcionalnosti

Sistem, ki smo ga v sklopu diplomske naloge implementirali (na sliki 3.1), uporabniku omogoča vodenje svojih delavniških dnevnikov (v nadaljevanju imenovanih *projektov*). Služil bo kot alternativa pisanju delavniških dnevnikov s pisarniškimi programi. Vsak projekt ima zbirko *korakov*. Korak opisuje posmezno dejavnost tehnološkega postopka, ki ga želimo zabeležiti z delavniškim dnevnikom.

Implementiran sistem sestavlja:

- mobilna aplikacija,
- glasovni pomočnik in
- strežnik.



Slika 3.1: Preprost načrt sistema

Osnovni uporabniški vmesnik sistema je mobilna aplikacija. Preko nje se uporabnik v sistem prijavi s svojim uporabniškim računom, ustvarja in odpira svoje delavniške dnevниke.

Delavniški dnevnik oz. projekt sestavljajo naslov, opis dela, opis možnih nevarnosti in seznam korakov postopka.

V odprt projekt lahko uporabnik, enega za drugim, dodaja korake, ki opisujejo potek dela. Posamezen korak sestavlja naslov, opis in trajanje. Uporabnik lahko tudi posname slike in jih doda v odprt projekt kot korak.

Glasovnega pomočnika smo uporabili kot dodaten uporabniški vmesnik, s katerim lahko uporabnik prostoročno dodaja korake v odprt projekt.

Z njim smo želeli doseči:

- prostoročno narekovanje besedilnega koraka projekta,

- prostoročno odpiranje obrazca za dodajanje koraka v projekt in
- prostoročno odpiranje kamere na mobilni napravi.

Podatki, ki jih uporabnik vnese preko mobilne aplikacije in glasovnega pomočnika, se hranijo na strežniku.

### **3.1.1 Smernice razvoja sistema za sestavljanje delavniškega dnevnika**

Pri načrtovanju sistema za pisanje opisov tehnoloških postopkov smo se osredotočali predvsem na nasledje točke:

- komponente sistema morajo biti enostavno zamenljive,
- uporabljene podatke (fotografije, zapiske) se mora hrani na enem mestu,
- sistem mora biti popolnoma funkcionalen brez glasovnega pomočnika.

Sistem, ki smo ga implementirali v tej raziskavi, služi kot primer kompleksnejšega sistema, katerega uporabnost bi radi izboljšali z uporabo glasovnega pomočnika.

Sistem mora podpirati ustvarjanje novega opisa tehnološkega postopka in odpiranje ter urejanje obstoječih opisov tehnološkega postopka.

Vsek opis tehnološkega postopka mora imeti naslov, opis, opis možnih nevarnosti pri delu in seznam korakov dela.

Seznam korakov dela mora podpirati dodajanje novih korakov, spreminjaњe vsebine obstoječih korakov, brisanje obstoječih korakov in spreminjanje vrstnega reda korakov.

### **3.1.2 Kaj smo želeli z glasovnim pomočnikom izboljšati?**

Glasovni pomočnik v našem sistemu služi kot uporabniški vmesnik za prostoročno dodajanje korakov v odprt projekt. Glasovni pomočnik je smiseln

način interakcije z računalniškim sistemom v situaciji, kjer uporabnik nima prostih rok ali je fizično oddaljen od naprave.

Osnovna funkcionalnost glasovnega pomočnika je bila prostoročno narekovanje besedila koraka v projektu. Poleg narekovanja besedila korakov smo implementirali odpiranje obrazcev za dodajanje korakov v projekt in zagon kamere.

Glasovni pomočnik v našem sistemu ima naslednje vloge:

- za trenutno odprt opis tehnološkega postopka je z njim možno dodajati korake postopka,
- z njim je možno odpreti kamero ali obrazec za dodajanje koraka.

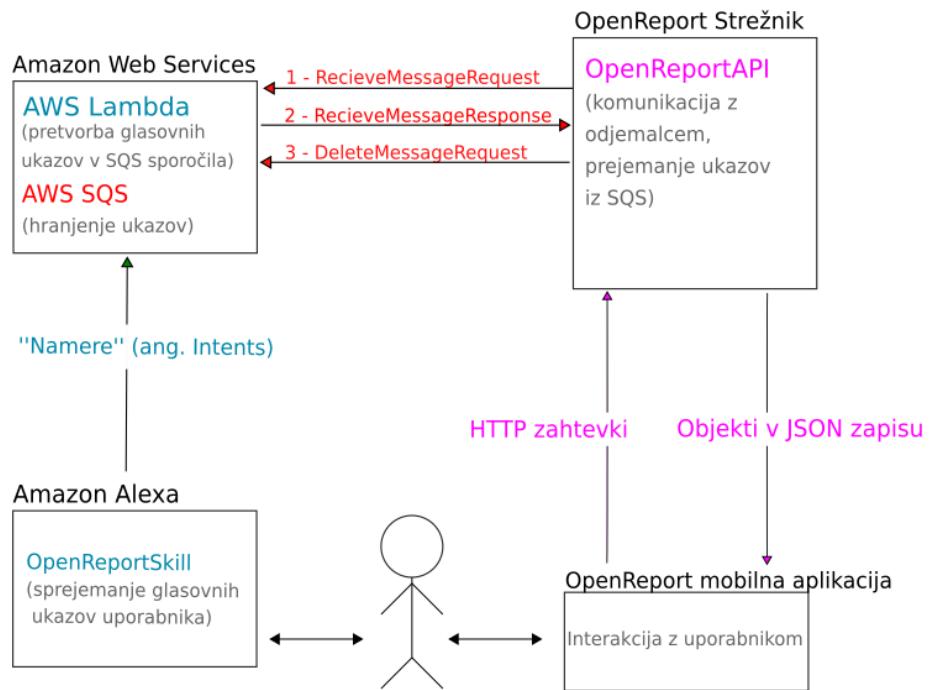
Raziskava iz leta 2018 [4] je pokazala izboljšano učinkovitost pri delu raziskovalcev v kemijskem laboratoriju, v katerega so integrirali glasovne pomočnike. Namen raziskave je bil preizkus praktične uporabnosti glasovnih pomočnikov za branje laboratorijskih postopkov in glasovno upravljanje laboratorijskih instrumentov. Pozitivni rezultati bi lahko bili ključnega pomena za slabovidne člane laboratoriijev. Kot glasovni pomočnik je bila uporabljena Amazon Alexa. Prepoznavanje govora in ukazov je bilo konsistentno in hitro, ne glede na spol uporabnika. Motnje pri razpoznavanju je povzročal večinoma ozadni hrup. Povprečna uspešnost prepoznavane ukazov je bila 95%. Raziskovalci so zabeležili tudi problem moteče kakofonije v laboratoriju, v katerem je več raziskovalcev, ki uporabljajo glasovni nadzor naprav.

### **3.2 Načrt sistema za sestavljanje opisov tehnoloških postopkov**

Sistem za sestavljanje opisov tehnoloških postopkov smo poimenovali OpenReport. Sistem OpenReport (na sliki 3.2) sestavlja:

- strežniški program,

- mobilna aplikacija,
- glasovni pomočnik.



Slika 3.2: Visokonivojski načrt sistema

Strežnik v podatkovni bazi hrani uporabnike, opise tehnoloških postopkov in korake. Uporabnik lahko do podatkov na strežniku dostopa preko odjemalca, ki s strežnikom komunicira preko API. V našem primeru je to mobilna aplikacija. Glasovni pomočnik smo uporabili kot dodatek k mobilni aplikaciji. Omogoča glasovno upravljanje aplikacije in narekovanje korakov. Preko mobilne aplikacije uporabnik lahko:

- opravi registracijo in prijavo,
- ustvari nov delavniški dnevnik,
- odpre obstoječe delavniške dnevниke,

- ustvarja in ureja korake delavnškega dnevnika,
- zajema slike in jih vstavlja v delavniški dnevnik,
- briše korake delavnškega dnevnika,
- ureja vrstni red korakov delavnškega dnevnika.

Za glasovnega pomočnika Amazon Alexa smo razvili „Skill“, s katerim uporabnik lahko:

- v odprto poročilo vstavi dobesedno narekovan korak,
- odpre obrazec za dodajanje novega besedilnega koraka,
- odpre kamero in obrazec za dodajanje koraka s fotografijo.

### 3.2.1 Zasnova aplikacije

Mobilna aplikacija bo služila kot osnoven način za interakcijo s strežnikom.

V tej diplomski nalogi preučujemo, ali je možno z vpeljavo glasovnega pomočnika v naš sistem izboljšati učinkovitost tega sistema. Da bodo rezultati raziskave čim bolj natančni, bomo dali učinkovitosti mobilne aplikacije velik poudarek. Uporabniški vmesnik bo zasnovan za čim hitrejše delo. To nam bo v zagotovilo, da uporabniški vmesnik aplikacije ni ozko grlo pri pišanju opisov tehnoloških postopkov.

## 3.3 Uporabljene tehnologije

### 3.3.1 .NET

.NET [1] je razvojna platforma, razvita s strani Microsofta. Obsega programske jezike, prevajalnike, orodja in knjižnice, ki omogočajo širok spekter primerov uporabnosti, hkrati pa se ohranja enovitost ozadne kode.

Tehnologije .NET ogrodja, ki smo jih uporabil v tej diplomski nalogi so:

- .NET Core - odprtokodna platforma za razvoj spletnih storitev,
- Xamarin - ogrodje za razvoj mobilnih aplikacij za najpogostejše mobilne operacijske sisteme (Android, iOS).

.NET smo izbrali, ker je zelo dobro integriran z Amazonovim AWS API-jem in je dobro dokumentiran.

### 3.3.2 Xamarin

Ogrodje Xamarin [14] je odprtokodno orodje za razvoj mobilnih aplikacij, ki ga je razvil Microsoft.

Z njim je mogoče deliti večino ozadne in ospredne kode med različnimi mobilnimi operacijskimi sistemi. Za ozadno kodo se uporablja .NET (C#), za ospredno kodo pa se uporablja XAML (Extensible Application Markup Language).

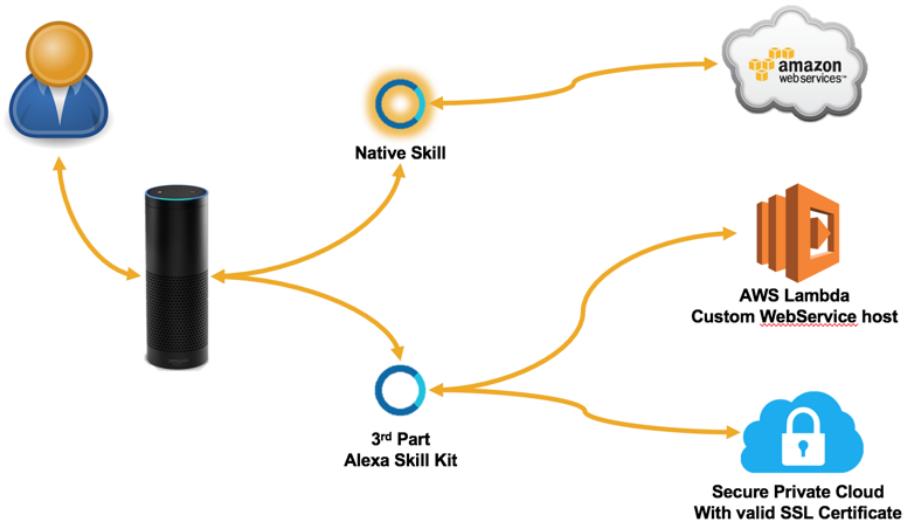
Xamarin smo izbrali, da bomo lahko strežniški program in mobilno aplikacijo programirali v enakem programskem jeziku (C#).

### 3.3.3 Amazon Alexa

// Besede ”Skill” ne bi prevajal, z enakim razlogom kot ne bi prevajal npr. Python. Gre za ime nekega produkta.

Amazon Alexa je glasovni pomočnik, razvit s strani podjetja Amazon [2]. Za Amazon Alexa smo se odločili, saj ponuja ogrodje za programiranje dodatnih funkcionalnosti, ki se imenuje Alexa Skills Kit. Poleg tega je Alexa enostavno integrirati z drugimi Amazonovimi spletnimi storitvami, ki jih bomo uporabili v tej diplomski nalogi (AWS SQS, AWS Lambda).

Alexine osnovne funkcionalnosti lahko nadgradimo s programi, ki se jim reče „Skill-i“ (na sliki 3.3) [12, 3]. Da lahko ustvarimo in objavimo „Skill“ rabimo račun Amazon razvijalca (*ang. Amazon Developer Account*).



Slika 3.3: Arhitektura Alexa „Skill-a“

„Skill“ sestavlja:

- **Poziv** - fraza, ki „Skill“ zažene,
- **Namera** - fraze, ki jih „Skill“ razpozna kot funkcije,
- **Krajišče** - omrežni vir, kjer se nahaja ozadna koda „Skill-a“.

Ko Alexa zasliši poziv ali katero od namer, glasovni posnetek pošlje na Amazonov Alexa Voice Service. Ta s pomočjo glasovnega razpoznavnega modela prepozna uporabnikove ukaze in pošlje poseben zahtevek na krajišče. To je lahko druga Amazonova storitev (npr. AWS Lambda), storitev na Microsoftovem Azure strežniku ali naš lasten strežnik, dostopen preko javne domene.

Ko krajišče obdela zahtevo, se odgovor pošlje nazaj na Alexa Voice Service v obliki znakovnega niza. Ta podatek se nato pošlje nazaj na Alexa, ki prejeti znakovni niz ”izgovori”.

### 3.3.4 Amazon Web Services

Amazon Web Services (v nadaljevanju AWS) je skupek oblačnih storitev, ki jih ponuja podjetje Amazon. Ponuja integracijo s pogostimi programskimi jeziki in ogrodji, kot so Java, .NET, Python in Node.js preko storitve AWS API. Pri tej diplomski nalogi smo se osredotočili na dva sistema iz skupka AWS.

#### Simple Queue Service

AWS Simple Queue Service (v nadaljevanju SQS) je sistem za pošiljanje teks-tovnih sporočil med odjemalci preko Amazonovih strežnikov [7]. Za hranjenje sporočil je potrebno registrirati "SQS Queue" (*slo. SQS vrsto*). Ta je lahko neurejena vrsta, kjer prejeta sporočila niso nujno urejena po času ustvarjanja, lahko pa je tipa FIFO (first in, first out), pri kateri je zagotovljeno, da dobimo vsa sporočila v vrstnem redu, v katerem so bila poslana. V sklopu te diplome smo uporabili vrsto tipa FIFO.

To storitev smo uporabili za komunikacijo med Amazon Alexa in OpenReport strežnikom.

#### Lambda

Ozadno kodo za Alexa „Skill“ smo gostili na platformi AWS Lambda, ki je storitev za gostovanje dogodkovno vodene ozadne kode [6]. Za to platformo smo se odločili zaradi dobre integracije z Alexa Skill Kit in razvojnim orodjem Visual Studio.

## 3.4 Strežnik

Za uporabo strežnika za hranjenje podatkov smo se odločili, da lahko do podatkov dostopamo iz različnih naprav preko enotnega vmesnika (API). To nam omogoča enostavno dodajanje odjemalcev in odpravi podvajanje

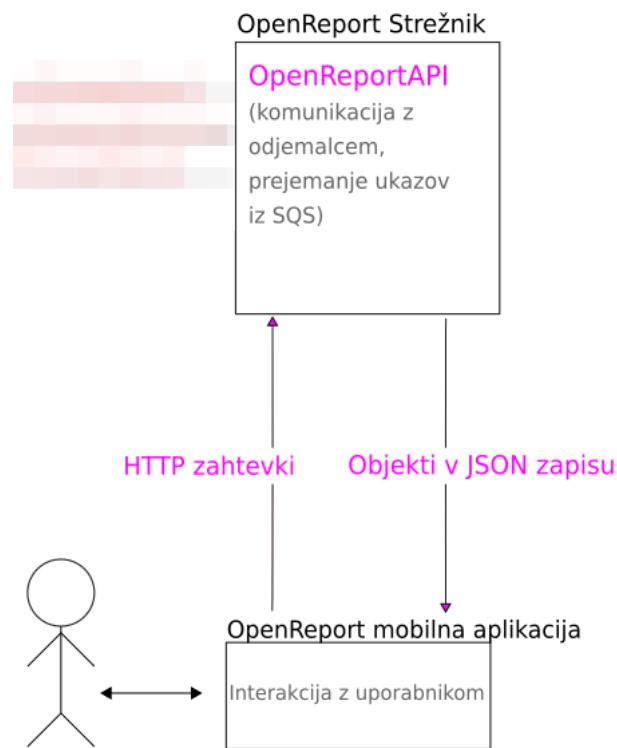
podatkov. Če bi podatke hranili na odjemalčevi napravi, bi bil dostop do njih iz druge naprave bistveno otežen.

V našem primeru bosta s strežnikom komunicirala mobilna aplikacija in glasovni pomočnik. Podatke bomo hranili v podatkovni bazi, ki jo streže SQL Server.

### 3.4.1 Komunikacija z odjemalci

Strežnik z odjemalci komunicira preko API. To metodo smo izbrali zaradi enostavnosti implementacije in možnosti širjenja nabora odjemalcev v prihodnosti.

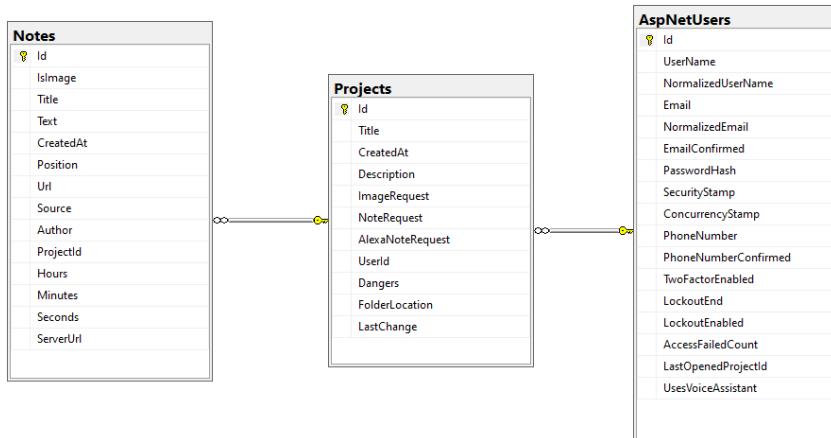
V našem primeru bo edini odjemalec mobilna aplikacija. Odjemalec na definirane funkcijске URL-je (slika 3.6) strežnika pošlje HTTP zahtevke. Če so zahtevki pravilno oblikovani, strežnik izvede predvideno funkcijo in rezultat te funkcije pošlje kot HTTP odgovor nazaj odjemalcu. Komunikacija med strežnikom in odjemalcem je prikazana z roza barvo na spodnji sliki (slika 3.4).



Slika 3.4: Komunikacija strežnika z odjemalcem

### 3.4.2 Podatkovni model

Podatkovna baza hrani tabele „AspNetUsers“, „Projects“ in „Notes“ (slika 3.5). Vsak uporabnik lahko ima 0 ali mnogo projektov (opisov tehnološkega postopka). Vsak projekt ima lahko 0 ali mnogo zapiskov (korakov).



Slika 3.5: ER model podatkovne baze

### 3.4.3 Implementacija API

API smo uporabili kot vmesnik za komunikacijo med odjemalcem in strežnikom, ki hrani podatke (slika 3.6). Zanj smo se odločili predvsem zaradi enostavnosti implementacije.

Ker lahko sistem uporablja več uporabnikov, smo se odločili implementirati sistema za avtentikacijo uporabnikov in avtorizacijo zahtev. Avtentikacija z uporabniškim imenom in gesлом omogoča preverjanje identitete. Ko se vzpostavi zaupanje, se prijavljenemu uporabniku dodeli avtorizacijski žeton. Iz žetona je možno razbrati, za katerega uporabnika gre in katere pravice so mu dodeljene.

Project	Users
GET /api/v1/projects	PUT /api/v1/users/requestava
POST /api/v1/projects	PUT /api/v1/users/releaseava
GET /api/v1/projects/{id}	POST /api/v1/users/current
DELETE /api/v1/projects/{id}	
GET /api/v1/projects/{id}/notes	GET /api/v1/connect
GET /api/v1/projects/{id}/nonotes	
POST /api/v1/projects/{id}/addnote	POST /api/v1/identity/register
POST /api/v1/projects/{id}/addimage	POST /api/v1/identity/login
POST /api/v1/projects/{id}/export/text	
POST /api/v1/projects/{id}/export/html	
PUT /api/v1/projects/last/closerequests	
PUT /api/v1/projects/{pid}/update/{nid}	
PUT /api/v1/projects/{pid}/{nid}/{positions}	
DELETE /api/v1/projects/{pid}/delete/{nid}	

Slika 3.6: Avtomatsko dokumentiranje funkcijskih URL-jev s Swagger

## Registracija in prijava

Vse operacije nad uporabnikovimi delavnískimi dnevniki morajo biti avtorizirane. Uporabnik sistema mora za uporabo sistema imeti registriran uporabniški račun. Na ta račun so vezani vsi njegovi delavníski dnevniki.

Preden dobi uporabnik dostop do svojih podatkov, mora vzpostaviti zupanje s strežnikom. To doseže s prijavo s pravilnim uporabniškim imenom in geslom. Če uporabnik še nima računa, se mora registrirati.

Pri registraciji mora odjemalec na strežnik poslati objekt razreda *RegisterUserRequest*. Odjemalec ga mora poslati na URL ”/identity/register”

preko metode POST. V objektu `RegisterUserRequest` se nahajata njegov e-mail naslov in geslo v čisti obliki.

```
RegisterUserRequest {
    string Email;
    string Password;
}
```

Ko strežnik prejme ta objekt, ga pošlje v avtentikacijsko storitev. V tej storitvi preveri, ali uporabnik s tem uporabniškim imenom že obstaja v bazi (je že bil registriran). Če obstaja, se zabeleži napaka in nadaljnja registracija se prekine. Če ta uporabnik ne obstaja, se kreira nov objekt razreda `User`. Polje `Password` se šifrira in se skupaj s poljem `Email` zapiše v ta objekt. Objekt se nato zapiše v podatkovno bazo v tabelo `Users`. Uporabnik pri registraciji dobi tudi svoj enoličen identifikator `UserID`.

V kolikor se je tokom avtentikacije dogodila kakršna koli napaka, se odjemalcu vrne objekt razreda `AuthFailedResponse`. V tem objektu se odjemalcu pošlje seznam vseh napak, ki jih je avtentikacijska storitev zabeležila pri neuspešni registraciji.

```
AuthFailedResponse {
    IEnumerable<string> Errors;
}
```

Če je bila registracija uspešna, se odjemalcu pošlje objekt razreda `AuthSuccessResponse`.

```
AuthSuccessResponse {
    string UserId;
    string Token;
}
```

V polje `Token` razreda `AuthSuccessResponse` se zapiše avtorizacijski žeton. Žeton je tipa JWT ali *JSON Web Token* [15]. Sestavlja ga e-mail uporabnika, uporabnikov enolični identifikator `UserID`, čas zapada žetona in tip simetričnega kodiranja, uporabljenega za enkripcijo žetona.

Objekt `AuthSuccessResponse` se pošlje nazaj odjemalcu. Odjemalec žeton iz prejetega objekta doda glavi vseh svojih HTTP zahtevkov na OpenReport strežnik. Strežnik ga uporabi pri poizvedbah po podatkovni bazi.

Prijava poteka podobno. Uporabnik mora poslati na strežnik objekt `LoginUserRequest`. Ciljni URL za ta zahtevek je ”/identity/login”. V objektu `LoginUserRequest` se nahajata njegov e-mail naslov in geslo.

```
(LoginUserRequest {  
    string Email;  
    string Password;  
})
```

Ko strežnik prejme ta objekt, ga pošlje v avtentikacijsko storitev. V tej storitvi preveri, ali uporabnik s tem e-mail naslovom že obstaja. Če uporabnik ne obstaja, ali pa je njegovo šifrirano geslo v podatkovni bazi drugačno kot to, kar je v polju `Password`, se zabeležijo napake in prijava se prekine. Odjemalec prejme objekt razreda `AuthFailedResponse`.

Če uporabnik obstaja in se njegovo šifrirano geslo iz polja `Password` ujema z gesлом v podatkovni bazi, je avtentikacija uspešna. Odjemalec prejme objekt razreda `AuthFailedResponse`.

V primeru uspešne prijave se avtorizacijski žeton `Token` generira in pošlje enako kot pri registraciji.

## Operacije z delavniškimi dnevnikami

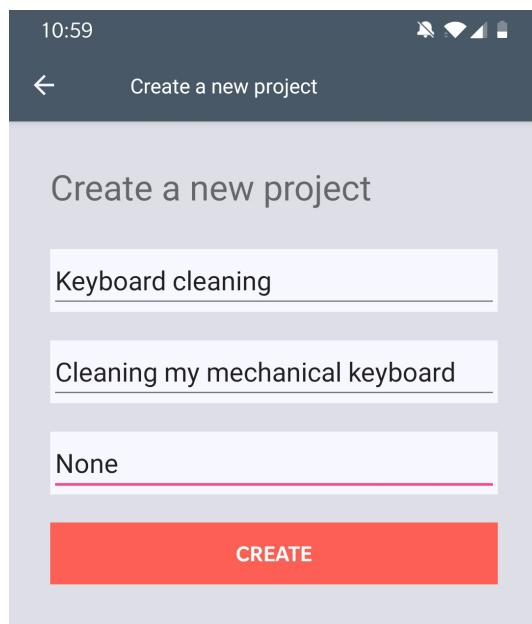
Avtenticiran uporabnik lahko dostopa do svoje zbirke delavniških dnevnikov. Vsak uporabnik lahko ima nič ali več delavniških dnevnikov.

Vse operacije nad uporabnikovim delavniškim dnevnikom morajo biti avtorizirane. V kolikor niso, bo API vedno javil napako `Bad Request: Not Authorised`.

Ko uporabnik želi ustvariti nov delavniški dnevnik (slika 3.7) (v nadaljevanju poglavja *projekt*), mora v mobilni aplikaciji podati naslov, kratek opis projekta in opis možnih nevarnosti.

Te podatke odjemalec zapiše v objekt razreda `CreateProjectRequest`.

```
CreateProjectRequest {  
    string Title;  
    string Description;  
    string Dangers;  
}
```



Slika 3.7: Ustvarjanje novega projekta v aplikaciji.

Ta objekt se nato pošlje preko POST metode na strežnik na naslov `/projects/create`. Storitev za operacije nad projekti nato kreira nov objekt razreda `Project` s podatki iz prejete zahteve. V primeru, da je zahteva ustrezno formulisana in avtorizirana, se projekt zapiše v podatkovno bazo. Odjemalcu se kot odgovor pošlje ustvarjen objekt tega projekta.

Razred `Project` izgleda tako:

```
Project {
```

```
int Id; // unikatni identifikator
string Title;
string Description;
string Dangers;
IEnumerable<Note> Notes; // seznam korakov
...
}
```

Uporabnik lahko dostopa do katerega koli svojega delavníškega dnevnika. Do njega dostopa tako, da pošlje avtoriziran GET zahtevek na URL ”/projects/{id}”. Polje {id} mora biti unikatni identifikator projekta.

Za izbris projekta mora uporabnik poslati DELETE zahtevo na URL ”/projects/{id}”. Če zahteva ni avtorizirana z ustreznim žetonom, se projekt ne izbriše.

### Operacije nad koraki delavníškega dnevnika

Korak delavníškega dnevnika opisuje eno od nalog v tehnološkem postopku, ki ga delavníski dnevnik opisuje. Vsak projekt ima lahko nič ali več korakov. Vsak korak ima naslov, opis in predvideno trajanje.

Korak je lahko izključno tekstovni, lahko pa ima tudi pripadajočo sliko.

Da uporabnik ustvari nov besedilni korak (na sliki 3.8 levo), mora v aplikaciji izpolniti obrazec za dodajanje koraka. Ta obrazec ima polja za naslov, besedilo in trajanje. Da uporabnik ta korak vnese, mora poslati objekt razreda Note, preko POST metode, na URL ”/projects/{id}/addnote”. Polje {id} mora biti unikatni identifikator projekta, katermu želi dodati korak. Razred Note izgleda tako.

```
Note {
    ...
    string Title;
    string Text;
    int Hours;
```

```
    int Minutes;  
    int Seconds;  
    ...  
}
```

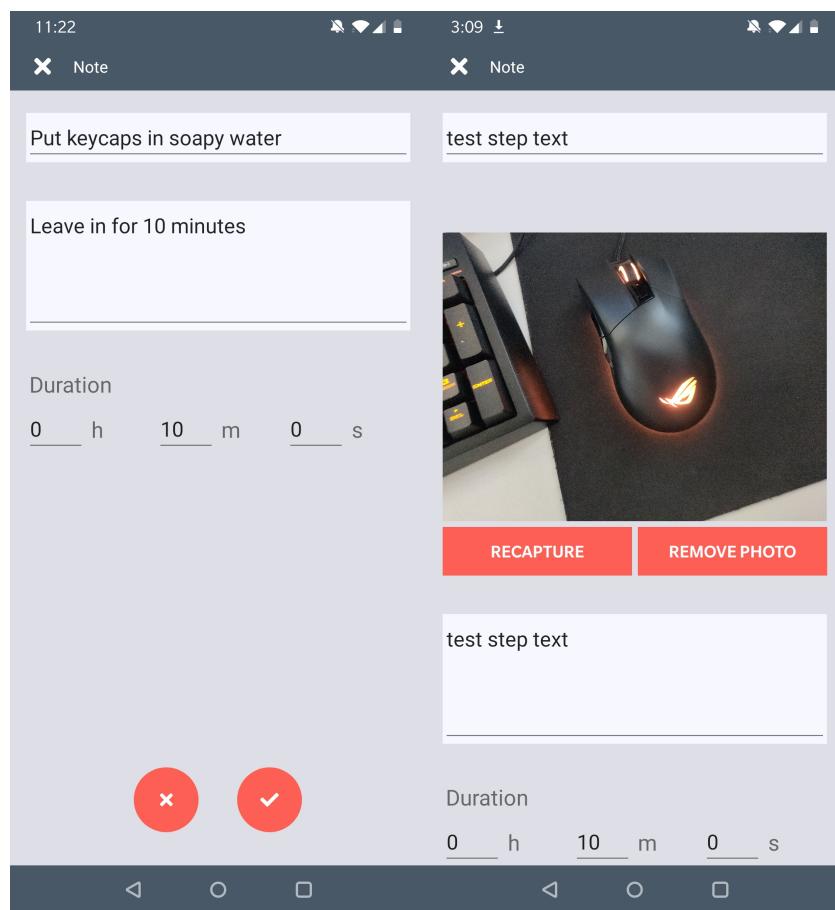
Naslov koraka se zapiše v polje Title, opis v polje Text, trajanje pa se zapiše v polja Hours, Minutes in Seconds.

Ko uporabnik pošlje objekt s podatki nas strežnik, strežnik najprej preveri, če je pošiljal tel poslal avtorizirano zahtevo. Iz avtorizacijskega žetona strežnik razbere uporabnikov `UserID`. Če je projekt z identifikatorjem `{id}` res v lasti uporabnika z identifikatorjem `UserID`, se bo korak dodal v zbirku korakov tega projekta.

Slikovni koraki (na sliki 3.8 desno) se v delavnški dnevnik dodajajo na podoben način. Uporabnik v mobilni aplikaciji odpre kamero in zajame fotografijo, ki se nanaša na korak, ki ga želi ustvariti. Ko jo zajame, se odpre obrazec za dodajanje naslova, opisa in trajanja koraka s fotografijo. Ko so podatki vnešeni, se slika zašifrira v znakovni niz po metodi Base64. Ostali podatki se zapišejo v objekt razreda `Note`.

Korak, zapisan v objekt razreda `Note` in v znakovni niz šifrirana slika se zapišeta v objekt razreda `UploadImageRequest`. Ta objekt se nato pošlje preko POST metode na URL `"/projects/{id}/addimage"`.

```
UploadImageRequest {  
    Note note;  
    string ImageString;  
}
```



Slika 3.8: Dodajanje korakov v aplikaciji.

V tem razredu predstavlja polje `ImageString` pripadajočo fotografijo, po metodi Base64 šfirirano v znakovni niz.

Za hranjenje tekstovnih in slikovnih korakov smo zaradi preprostosti implementacije uporabili isti razred (`Note`). Tekstovni in slikovni korak se ločita v vrednosti boolean zastavice `IsImage`. Tekstovni korak ima to zastavico nastavljeno na vrednost `false`, slikovni pa `true`.

```
Note {  
    bool IsImage;  
    string Title;  
    string Text;
```

```

int Hours;
int Minutes;
int Seconds;
string Url; // Lokacija slike na odjemalcu
string ServerUrl; // Lokacija slike na strežniku
...
}

```

Poljema `Url` in `ServerUrl` se dodeli vrednost samo pri slikovnih korakih. Ko se na odjemalcu zajame fotografija, odjemalec nastavi vrednost polja `Url` na lokacijo ustvarjene fotografije v datotečnem sistemu.

Polje `ServerUrl` se nastavi šele na strežniku. Ko strežnik prejme zahtevo za kreiranje slikovnega koraka, se korak `Note` prebere iz objekta `UploadImageRequest` in zapiše v podatkovno bazo. Nato se `ImageString` dekodira v slikovno datoteko in hrani na strežniku. Ko se fotografija uspešno zapiše v datotečni sistem, se v polje `ServerUrl` zapiše lokacija pravkar ustvarjene datoteke. Spremembe se hranijo v podatkovni bazi.

### **Urejanje in brisanje in korakov v projektu**

Uporabnik lahko korake, ki pripadajo njegovim delavnškim dnevnikom spreminja. Spreminja lahko njihovo vsebino, naslov in trajanje.

Po končanem urejanju besedilnih korakov, mora odjemalec poslati avtorizirano PUT zahtevo na URL ”/projects/{pid}/update/{nid}”. Telo zahteve mora vsebovati objekt razreda `Note`, ki ga je uporabnik spremenil.

Pri spremnjanju vsebine slikovnih korakov mora odjemalec poslati avtorizirano PUT zahtevo na naslov ”/projects/{pid}/update/{nid}/image”. Telo te zahteve mora vsebovati objekt razreda `UploadImageRequest`. V objektu `UploadImageRequest` mora biti polje `Note` objekt, ki ga želimo posodobiti. Polje `ImageString` mora biti šifrirana slika, ki bo zamenjala prejšnjo sliko. Slika mora biti šifrirana po metodi Base64.

Pri izbrisu koraka iz projekta, mora uporabnik poslati avtorizirano zahtevo na URL ”/projects/{pid}/delete/{nid}”. V tem primeru je polje

{pid} unikatni identifikator projekta, v katerem se nahaja korak, {nid} pa unikatni identifikator objekta Note, ki ga želimo izbrisati. V kolikor ima ta objekt Note postavljeno zastavico IsImage na true, se poleg zapisa v bazi izbriše tudi pripadajoča slikovna datoteka.

### Spreminjanje vrstnega reda korakov v projektu

Uporabnik lahko korakom v svojih projektih spreminja vrstni red prikaza. Izbran korak lahko zamakne za poljubno število mest proti začetku ali koncu seznama. Položaj koraka Note v seznamu lahko razberemo iz atributa Position. Prvi korak ima Position 0, drugi 1, itd.

```
Note {
    int Id;
    int Position;
    bool IsImage;
    string Title;
    string Text;
    int Hours;
    int Minutes;
    int Seconds;
    string Url;
    string ServerUrl;
}
```

Pri dodajanju korakov v projekt se obstoječe korake projekta razvrsti po vrednosti polja Position. Najdemo največjo vrednost tega polja, ki se v seznamu pojavi in ji prištejemo 1. Nato to vrednost priredimo polju Position novo kreiranega koraka.

Ko uporabnik želi spremeniti pozicijo koraka v projektu, moramo poslati zahtevo PUT na URL ”/projects/{pid}/{nid}/{positions}”.

Polje {pid} predstavlja unikatni identifikator projekta, v katerem se nahaja korak. Polje {nid} predstavlja unikatni identifikator koraka Note. Po-

lje `{positions}` pa prestavlja število mest za kolikor ga želimo prestaviti. To število je lahko pozitivno ali negativno celo število. Negativna vrednost `{positions}` prestavi korak proti začetku seznama, pozitivna pa proti koncu.

## Izvoz projektov

Projekt lahko uporabnik izvozi v dva formata, v besedilno datoteko in HTML dokument.

V besedilno datoteko ga uporabnik izvozi, tako, da pošlje avtorizirano GET zahtevo na URL `"/projects/{id}/export/text"`. V HTML dokument ga uporabnik izvozi, tako, da pošlje avtorizirano GET zahtevo na URL `"/projects/{id}/export/html"`.

Pri izvozu v besedilno datoteko, storitev za upravljanje projektov na strežniku ustvari novo besedilno datoteko. Najprej vanjo vpiše naslov in opis projekta ter možne nevarnosti pri delu. Nato uredi korake po vrednosti stolpca **Position** in besedila korakov eno za drugim zapiše v datoteko. Pri slikovnih korakih se, poleg besedila, zapiše tudi lokacija pripadajoče fotografije.

Pri izvozu v HTML dokument (slika 3.9) se naslov projekta zapiše kot HTML naslov H1, opis projekta kot naslov H2, koraki pa kot HTML odstavki. V HTML dokumentu lahko prikažemo poleg besedila slikovnih korakov tudi slike.

( file:///C:/Users/Public/Documents/OpenReport/0\_Keyboard cleaning/20200811\_21\_02\_34\_Keyboard\_cleaning.html)

## Keyboard cleaning

**Description**

Cleaning my mechanical keyboard

**Dangers**

None

**1. Prepare the table and tools**

Est. duration: 0 h 0 min 0 seconds

**2. Remove keycaps**

Use the tool, provided with the keyboard  
Est. duration: 0 h 10 min 0 seconds

**3. Put keycaps in soapy water**

Leave in for 10 minutes  
Est. duration: 0 h 10 min 0 seconds

**4. Wipe keycaps until they are dry**

Est. duration: 0 h 20 min 0 seconds

**5. Place keycaps in their places**



Est. duration: 0 h 5 min 0 seconds

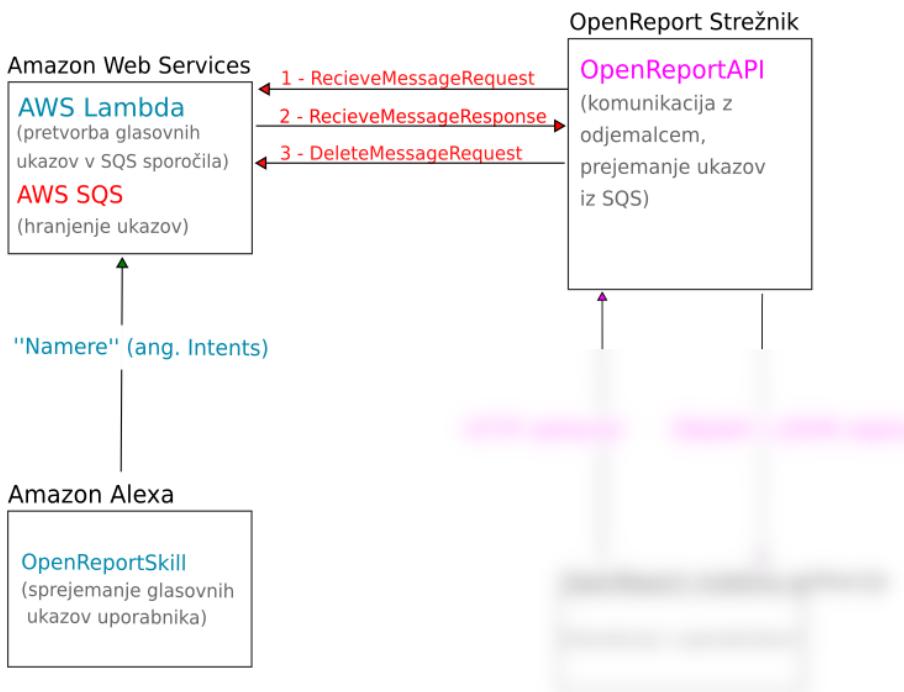
Slika 3.9: Projekt, izvožen v HTML dokument

Lokacija izvožene datoteke se nahaja v polju `FolderLocation` v razredu `Project`. Privzeta lokacija, ki se nastavi ob kreiranju projekta je ”`C:/users/$USER/Public Documents/OpenReport`”.

```
Project {  
    int Id;  
    string Title;  
    string Description;  
    string Dangers;  
    string FolderLocation; // lokacija pripadajočih datotek
```

```
IEnumerable<Note> Notes;
...
}
```

### 3.4.4 Komunikacija strežnika z glasovnim pomočnikom



Slika 3.10: Komunikacija med strežnikom in glasovnim pomočnikom preko SQS

Strežnik z glasovnim pomočnikom Amazon Alexa komunicira preko storitve Simple Queue Service (SQS) (slika 3.10) iz sklopa storitev Amazon Web Services.

Glasovni pomočnik od uporabnika prejme izgovorjen glasovni ukaz. Na podlagi tega ukaza, glasovni pomočnik na SQS vrsto odloži tri različna besedilna sporočila:

- „addnote“,
- „addimage“,
- „{narekovano besedilo}“,

Strežnik periodično prejema sporočila iz SQS vrste. Ko sporočilo prejme, na podlgi njegove vsebine izvede eno od treh možnih funkcij.

- Sporočilo „addnote“ sporoči strežniku, naj na odjemalcu odpre obrazec za dodajanje besedilnega koraka.
- Sporočilo „addimage“ sporoči strežniku, naj na odjemalcu odpre kamero in obrazec za dodajanje slikovnega koraka.
- Sporočilo „{narekovano besedilo}“ sporoči strežniku, naj vsebino tega sporočila doda odprtemu projektu kot tekstovni korak.

V sporočilu „{narekovano besedilo}“ se nahaja besedilo, ki ga je glasovni pomočnik razpoznał kot dobesedni narek koraka. Primer takšnega sporočila je ”this is a literal dictation“ ali ”prepare the table“.

### Utemeljitev uporabe SQS

Prenos sporočil med Amazon Alexo in strežnikom smo realizirali z uporabo storitve AWS SQS. Pri uporabi SQS je prenos podatkov resda manj učinkovit kot pri direktni povezavi preko spletne vtičnice (*ang. Web Socket*).

Glavni faktor, zaradi katerega smo se odločili za uporabo AWS SQS, je, da z dodatno plastjo za hranjenje sporočil, dosežemo enostavno zamenljivost glasovnega pomočnika.

Pri morebitni menjavi glasovnega pomočnika, strežniškega programa ne bi spremnjali. Potrebno bi bilo le implementirati program za novega glasovnega pomočnika, da na AWS SQS odlaga sporočila v strežniku razumljivem formatu („addnote“, „addimage“,...).

### Zahtevanje in izpust glasovnega pomočnika

Na enem OpenReport strežniku bomo uporabljali samo enega glasovnega pomočnika. Tega pomočnika lahko uporabnik ”zahteva” za lastno uporabo in ga po uporabi ”izpusti”. Če ima določen uporabnik nase vezanega pomočnika, vidimo po vrednosti zastavice `UsesVoiceAssistant`. Ta se nahaja v uporabnikovem objektu razreda `User`. Uporabnik pomočnika uporablja, če je zastavica nastavljena na `True`.

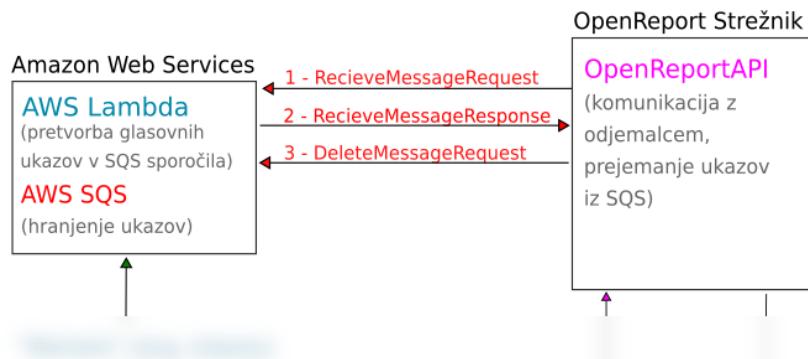
Ko uporabnik odpre katerega od svojih projektov, se identifikator tega projekta nastavi v polje `LastOpenedProjectId` v njegovem objektu razreda `User`.

Zahtevki, ki jih strežniku pošlje Alexa preko SQS vrste, se navezujejo na projekt s tem identifikatorjem.

```
User : IdentityUser {  
    string Id;  
    string Email;  
    string Password;  
    IEnumerable<Project> Projects;  
    ...  
    int LastUsedProjectId;  
    bool UsesVoiceAssistant;  
}
```

### Delovanje

Komunikacija med strežnikom in glasovnim pomočnikom je prikazana z rdečo barvo na spodnji sliki (slika 3.11).



Slika 3.11: Komunikacija med strežnikom in SQS

Na strežniku OpenReport teče storitev za komunikacijo z glasovnim pomočnikom. Ta storitev vsakih pet sekund na SQS pošlje zahtevek **RecieveMessageRequest**.

```

RecieveMessageRequest {
    AttributeName
    MaxNumberOfMessages
    QueueUrl
    WaitTimeSeconds
    ...
}
    
```

AWS SQS kot odgovor vrne objekt tipa **RecieveMessageResponse**.

```

RecieveMessageResponse {
    IEnumerable<Message> Messages;
    ...
}
    
```

V tem odgovoru se nahaja seznam sporočil, ki čakajo na sprejem iz SQS vrste. V kolikor je v odgovoru vsaj eno sporočilo, pregledamo telo vseh sporočil.

Ko strežnik sporočila sprejme, jih mora izbrisati iz SQS vrste. Če jih sproti ne izbriše, iz FIFO vrste ne more brati najnovejših sporočil, ampak le 5 najstarejših.

Sporočila izbriše iz SQS vrste z zahtevkom `DeleteMessageRequest`. Ta zahtevek hrani referenco na sporočilo, ki ga želimo izbrisati iz SQS vrste. AWS SQS storitev v odgovor vrne `DeleteMessageResponse`, a v tej diplomski nalogi tega odgovora ne uporabimo več.

Zahtevki, ki jih strežnik prejme od glasovnega pomočnika se odražajo na zadnjem projektu, ki ga uporabnik z glasovnim pomočnikom odpre.

- Zahtevek ”`addnote`” nastavi zastavico ”`NoteRequest`” na `true`,
- zahtevek ”`addimage`” nastavi zastavico ”`ImageRequest`” na `true`,
- zahtevek ”`{narekovano besedilo}`” nastavi zastavico ”`AlexaNoteRequest`” na `true` in prejeto besedilo doda projektu kot korak.

```
Project {
    int Id;
    string Title;
    string Description;
    string Dangers;
    string FolderLocation;
    IEnumerable<Note> Notes;
    bool NoteRequest;
    bool ImageRequest;
    bool AlexaNoteRequest;
}
```

Če je vsebina prejetega sporočila enaka `addnote`, storitev preveri, kateri uporabnik ima trenutno nase vezanega glasovnega pomočnika. Nato v projektu, ki ga je nazadnje odprl, nastavi vrednost zastavice `NoteRequest` na `true`.

Če je vsebina enaka ”`addimage`”, storitev preveri, kateri uporabnik ima trenutno nase vezanega glasovnega pomočnika. Nato v projektu, ki ga je nazadnje odprl, nastavi vrednost zastavice `ImageRequest` na `true`.

Če vsebina prejetega sporočila ni enaka ”`addimage`” ali ”`addnote`”, pomeni, da je prejeto sporočilo dobesedno narekovan korak. Kot prej, storitev preveri, kateri uporabnik ima nase vezanega glasovnega pomočnika. Nato v projektu, ki je nazadnje odprl, nastavi vrednost zastavice `AlexaNoteRequest` na `true`. Poleg tega ustvari nov objekt razreda `Note`, ki ima naslov ”Voice note”, vsebina pa je telo prejetega sporočila. Nov objekt se nato vstavi v zadnji odprt projekt.

### 3.5 Implementacija Alexa „Skill-a“

Alexa „Skill“ je program, ki glasovnemu pomočniku Amazon Alexa doda funkcionalnosti. Alexo in zasnovan „Skill“ smo uporabili kot dodaten uporabniški vmesnik za upravljanje z mobilno aplikacijo.

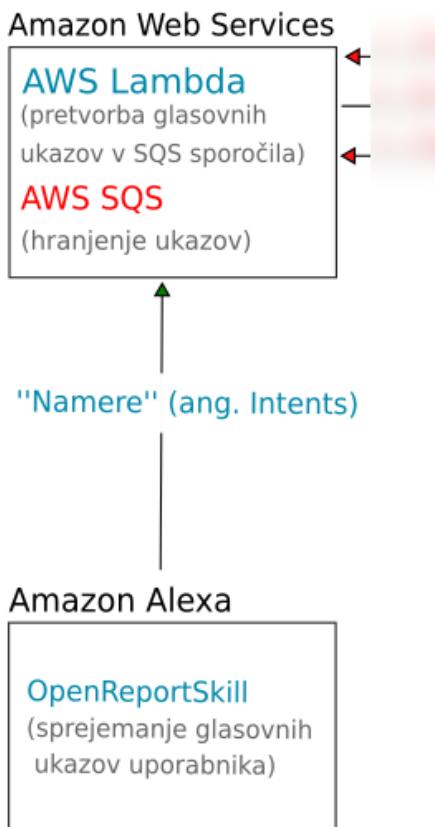
Z njim smo želeli doseči:

- prostoročno narekovnaje besedilnega koraka,
- prostoročno odpiranje obrazca za dodajanje korakov v odprt projekt in
- prostoročno odpiranje kamere in dodajanje slikovnega koraka v projekt.

Ko Alexa zasliši poziv „*make a report note*“, se naš „Skill“ začne izvajati.

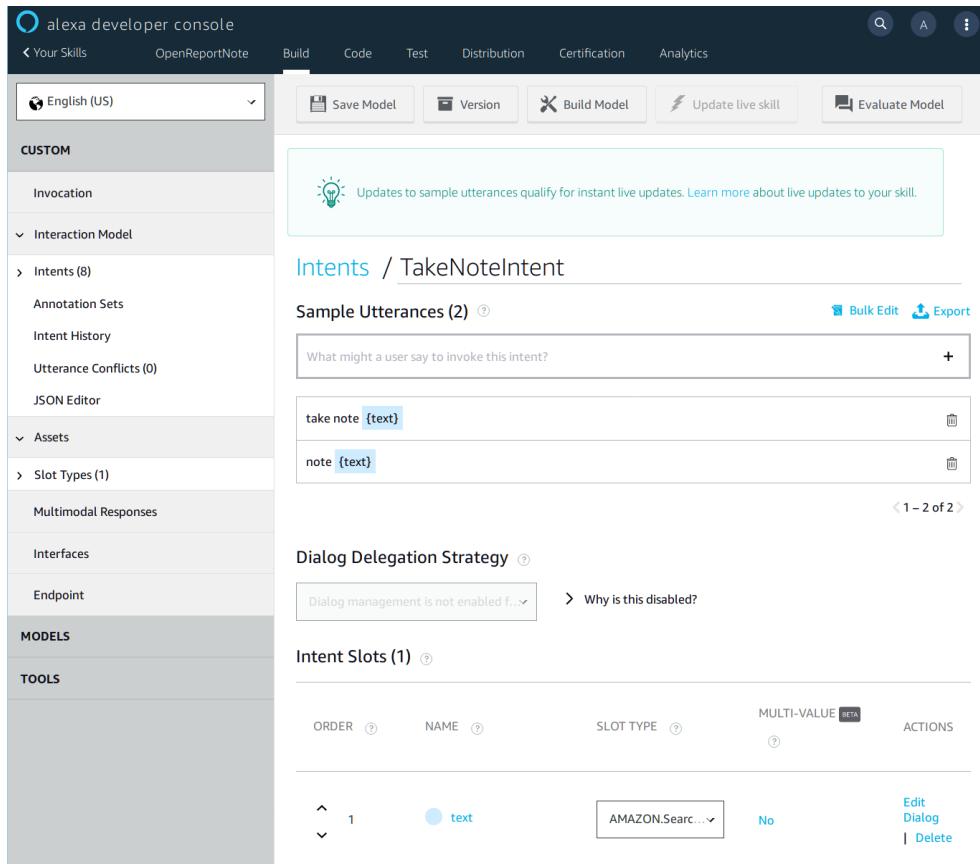
Vsak nadaljnji ukaz, ki ga uporabnik izreče, preden se „Skill“ preneha izvajati, se bo primerjal s frazami namer. Namere se uporabljajo za zagon funkcij, ki smo jih implementirali v ozadni kodi „Skill-a“ (slika 3.12). V našem „Skill-u“ smo definirali tri glavne namere:

- `TakeNoteIntent`,
- `OpenTextNoteFormIntent`,
- `OpenImageNoteFormIntent`.



Slika 3.12: Načrt komunikacije Alexe, AWS Lambda in AWS SQS

`TakeNoteIntent` (slika 3.13) se zažene, ko po pozivu uporabnik izreče ”`take note {besedilo}`” ali ”`note {besedilo}`”. Polje `{besedilo}` je besedilna spremenljivka, v katero „Skill“ hrani razpoznano besedilno vsebino koraka. V primeru, da uporabnik izreče „`note unscrew the backplate`“, je vrednost spremenljivke `{besedilo}` „`unscrew the backplate`“.



Slika 3.13: TakeNoteIntent v Alexa Developer nadzorni plošči

**TakeNoteIntent** v ozadni kodi „Skill-a“ zažene funkcijo, ki bo ustvarila novo SQS sporočilo. V telo tega sporočila funkcija vstavi vrednost spremenljivke `{besedilo}` in ga pošlje v SQS vrsto. To SQS sporočilo prebere strežnik in ga kot korak vstavi v odprt projekt. „Skill“ zatem vrne uporabniku odgovor ”Noted!“ in se preneha izvajati.

**OpenTextNoteFormIntent** se zažene, ko po invokacijski frazi uporabnik izreče ”create a text note”.

**OpenTextNoteFormIntent** v ozadni kodi „Skill-a“ zažene funkcijo, ki ustvari novo SQS sporočilo. V telo tega sporočila funkcija vstavi besedilo ”addnote“ in ga pošlje v SQS vrsto. To SQS sporočilo prebere strežnik in mobilni aplikaciji sporoči, naj odpre obrazec za dodajanje novega besedilnega

koraka. „Skill“ zatem vrne uporabniku odgovor ”Opening the form!” in se preneha izvajati.

### [Intents / OpenTextNoteFormIntent](#)

---

#### Sample Utterances (2) ?

What might a user say to invoke this intent?

create a text note

add a text note

Slika 3.14: OpenTextNoteFormIntent

`OpenImageNoteFormIntent` se zažene, ko po pozivu uporabnik izreče ”take a picture”.

Ta namera v krajišču „Skill-a“ zažene funkcijo, ki ustvari novo SQS sporočilo. V telo tega sporočila funkcija vstavi besedilo ”addimage” in ga pošlje v SQS vrsto. To SQS sporočilo prebere strežnik in mobilni aplikaciji sporoči, naj odpre kamero. „Skill“ zatem vrne uporabniku odgovor ”Launching camera!” in se preneha izvajati.

---

### [Intents / OpenImageNoteFormIntent](#)

---

#### Sample Utterances (1) ?

What might a user say to invoke this intent?

take a picture

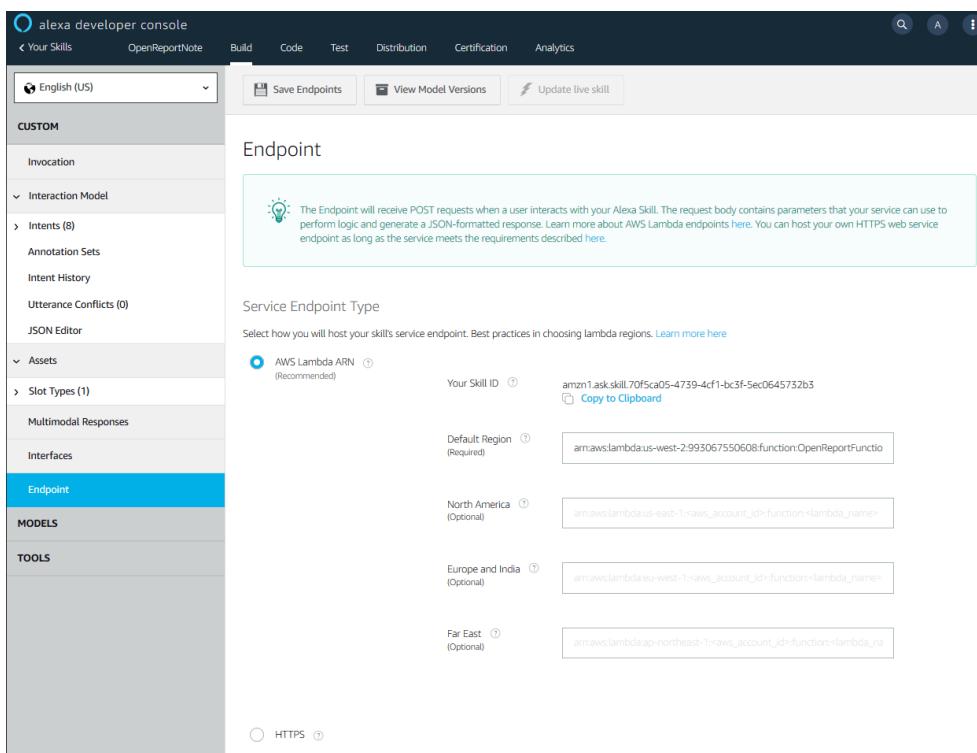
Slika 3.15: OpenImageNoteFormIntent

### 3.5.1 Izvajanje „Skill-a“ po korakih

Alexa „Skill“ se začne izvajati, ko uporabnik izreče poziv „*make a report note*“. Alexa posnet govorni ukaz pošlje na Amazon Alexa Server. Tam se s pomočjo NLU razpoznavnega modela poskusi pretvoriti v znakovni niz. Ta znakovni niz se primerja z vsemi definiranimi invokacijskimi frazami za „Skill-e“, ki so vezani na naš Amazon račun.

Če Amazon Alexa Server uporabnikov glasovni ukaz razpozna kot „*make a report note*“, se začne izvajati OpenReportAlexaSkill. V krajišče „Skill-a“ se pošlje zahteva tipa `LaunchRequest`.

Krajišče nastavimo v Alexa Developer nadzorni plošči (slika 3.16). Krajišče je lahko AWS Lambda funkcija, lahko pa je naš lasten strežnik, dostopen preko javne domene in zaščiten z SSL. Krajišče je v našem primeru gostovan na storitvi AWS Lambda. Ob prejetju `LaunchRequest` krajišče našega „Skill-a“ vrne vprašanje ”What now?“.

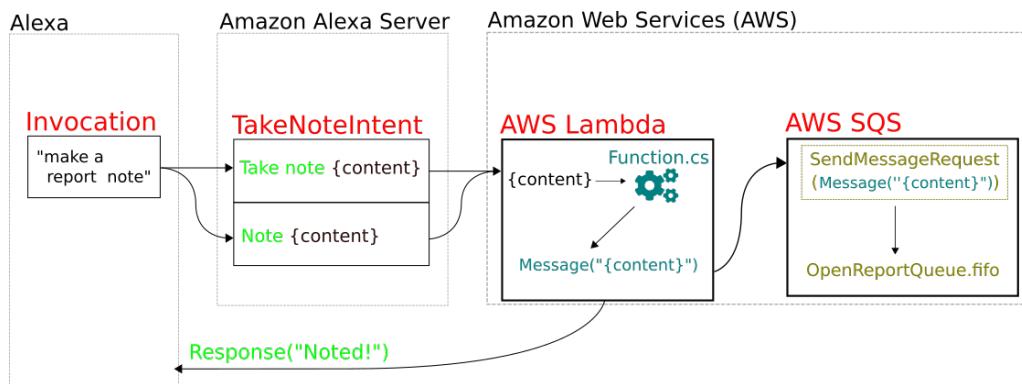


Slika 3.16: Krajišče OpenReportAlexaSkill-a v Alexa Developer nadzorni plošči

Naslednji glasovni ukaz, po zagonu „Skill-a“, se primerja s frazami za namere. Če se ukaz ujema z frazami, ki zaženejo katero od namer TakeNoteIntent, OpenTextNoteFormIntent ali OpenImageNoteFormIntent, se na krajišče poslje zahteva tipa IntentRequest.

Iz prejetega IntentRequest-a nato dobimo ime namere. Na podlagi imena prejete namere ločimo, ali gre za TakeNoteIntent, OpenTextNoteFormIntent ali OpenImageNoteFormIntent in zaženemo funkcije, opisane v prejšnjem poglavju.

Oris obdelave poziva „Skill-a“ in namere TakeNoteIntent je prikazan s spodnjo sliko (slika 3.17).



Slika 3.17: Načrt Alexa Skilla

### 3.5.2 Ovire pri izdelavi Alexa „Skill-a“

Pri implementaciji Alexa „Skill-a“ smo naleteli na največ težav v sklopu te raziskave.

Prva je bila jezikovne narave, saj Alexa ne podpira prepoznavanja slovenskega jezika. Zato smo se odločili za uporabo angleškega jezika.

Naslednja težava je bila definicija pozivne fraze. Fraze, kot so ”take a note” ali ”take a picture” so že rezervirane v sklopu Alexinih privzetih funkcionalnosti. To pomeni, da jih za naš „Skill“ ne moremo uporabiti.

Izogibati smo se morali tudi frazam, ki so bile rezerviranim frazam podobne (npr. ”open report note” itd.). Tudi take fraze so se velikokrat razpoznale kot rezervirane fraze.

Težavo je predstavljala izbira poziva, ki je bil hkrati kratek, intuitiven in se ni napačno interpretiral v poziv privzetih funkcionalnosti. Poziv ”make a report note” je bila najboljši kompromis, ki smo ga našli.

Težavo je predstavljalo tudi razpoznavanje dolgih stakov, še posebej, če se ti niso ujemali z definiranimi frazami namer. To je zelo omejilo možnost narekovanja besedila korakov s pomočjo Amazon Alexe.

Naslednjo težavo je predstavljal čas, ki ga je Alexa porabila za interpretacijo definiranih glasovnih ukazov. Za procesiranje glasovnih ukazov je porabila v povprečju od 3 do 6 sekund.

## 3.6 Implementacija mobilne aplikacije

Mobilno aplikacijo smo uporabili kot primer odjemalca za naš strežnik (slika 3.4).

Do strežnika dostopa preko API vmesnika. Zahtevki se prenašajo preko HTTP protokola.

Mobilno aplikacijo smo se odločili razviti s tehnologijo Xamarin. Razlogi za to so predvsem naše dobro poznavanje tehnologije in enostavna integracija z ostalimi tehnologijami iz .NET sklopa.

### 3.6.1 Pristop

Pristop razvoja aplikacije, ki smo ga uporabili za programiranje aplikacije, se imenuje "Model View View-Model" (v nadaljevanju MVVM). Pri tem pristopu aplikacijo razdelimo na tri dele (slika 3.18).

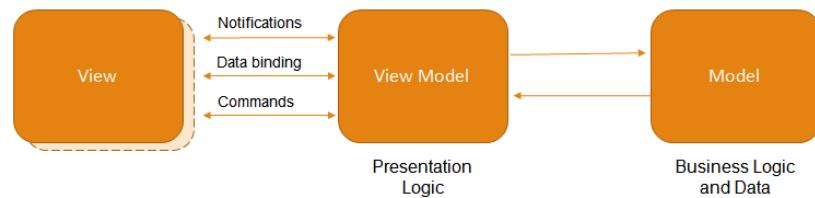
**Model** je del, kjer definiramo elemente naše "poslovne logike" (opis tehnološkega postopka, korak, uporabnik,...).

**View** je uporabniški vmesnik, ki ga vidi uporabnik.

**ViewModel** pa se uporablja, da se poveže funkcije uporabniškega vmesnika in modela ("poslovne logike") ter po potrebi preoblikuje podatke.

Rezultat upoštevanja tega pristopa je čista koda, ki nima prepleteneih elementov med ozadno kodo in uporabniškim vmesnikom. "Model" vsebuje le abstrakcijo naših podatkov in "poslovno logiko". Ti podatki se v "ViewModel-u" pretvorijo v obliko, ki bo prikazana uporabniku. "View" pa nato servira pripravljene podatke uporabniku v obliki grafičnih elementov.

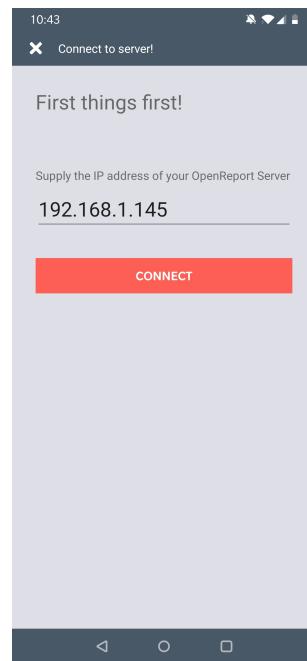
Na spodnji sliki lahko vidimo oris pristopa MVVM (slika 3.18) [11].



Slika 3.18: Shema pristopa MVVM

### 3.6.2 Povezava na strežnik, prijava in registracija uporabnika

Ob zagonu mobilne aplikacije, uporabnika pričaka stran za povezavo na OpenReport strežnik. V besedilno polje mora vpisati IP naslov strežnika. Ta naslov bomo v nadaljevanju podpoglavlja označevali z **naslovstrežnika**.



Slika 3.19: Povezavna stran

Za kreiranje in avtorizacijo HTTP zahtevkov v okviru aplikacije skrbi sto-

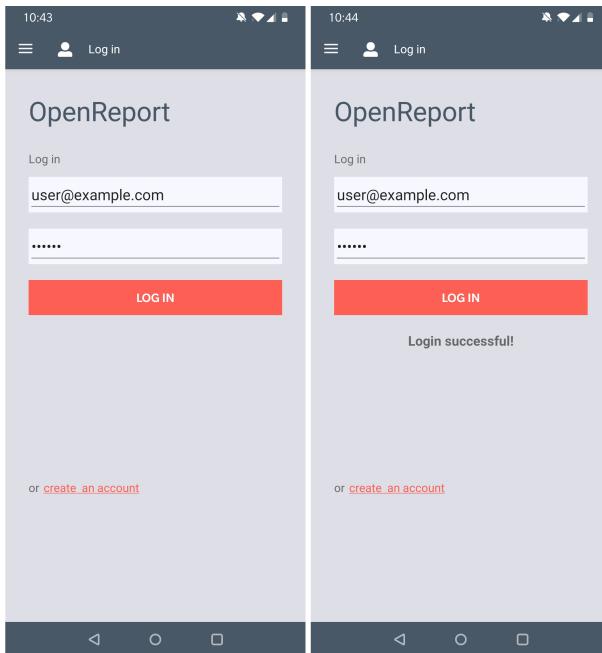
ritev za komunikacijo s strežnikom (razred `OpenReportServerCommunicationService.cs`). Storitev se registrira ob zagonu aplikacije po metodi Dependency Injection. Za to metodo registracije storitev smo se odločili, saj omogoča enostavno uporabo instanc registriranih storitev kjerkoli v aplikaciji.

Ko uporabnik pritisne na gumb "Connect", se preveri, ali je naslov pravilno formuliran. Če je, storitev za komunikacijo s strežnikom pošlje neavtorizirano GET zahtevo na "**naslovstrežnika/connect**". Če je na tem naslovu res OpenReport strežnik, se kot odgovor odjemalcu pošlje boolean vrednost `true`.

Storitev za komunikacijo s strežnikom si nato hrani vrednost **naslovstrežnika** kot predpono vseh nadaljnjih zahtevkov.

Ob neuspešnem poskusu povezave, se uporabniku na strani prikaže opis napake. Če je poskus uspešen, se stran za povezavo na strežnik zapre.

Prikaže se stran za prijavo. Na tej strani uporabnik vpiše svoje uporabniško ime ali geslo, lahko pa odpre stran za registracijo računa.



Slika 3.20: Prijavna stran pred poskusom prijave (levo), in po uspešnem poskusu prijave (desno)

Za prijavo in registracijo skrbi avtentikacijska storitev (razred `IdentityService.cs`). Ta storitev je, enako kot storitev za komunikacijo s strežnikom, registrirana po metodi Dependency Injection. Uporabljamo jo lahko, ko je storitev za komunikacijo uspešno vzpostavila komunikacijo s strežnikom.

Poskus prijave poteka tako, avtentikacijska storitev uporabniško ime in geslo zapiše v objekt razreda `LoginUserRequest`. Uporabniško ime in geslo se vneseta na prijavni strani.

```
(LoginUserRequest {
    string Email;
    string Password;
})
```

Nato se objekt `LoginUserRequest` pošlje na strežnik. Odjemalec ga pošlje na URL ”/identity/login” preko POST metode. Strežnik ob uspešni

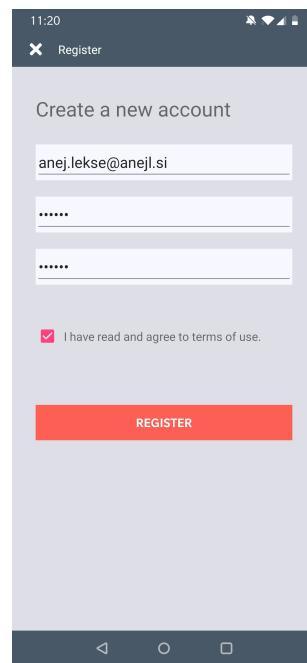
avtentikaciji vrne objekt razreda `AuthSuccessResponse`. Iz tega objekta uporabnik prebere svoj avtorizacijski žeton. Ta žeton hrani avtentikacijska storitev. Komunikacijska storitev ta žeton doda v glavo nadalnjih HTTP zahtevkov.

Če je avtentikacija neuspešna, strežnik vrne objekt razreda `AuthFailedResponse`, v katerem je podan opis napake. Ta opis napake se nato uporabniku izpiše na prijavni strani.

Stran za registracijo (slika 3.21) je podobna strani za prijavo, le, da ima dve vnosni polji za vpis gesla. Vrednost v obeh poljih za geslo mora biti enaka, da se lahko uporabnik registrira.

Registracijski zahtevek se pošlje na strežnik na URL ”/identity/register”. Zahtevek je objekt razreda `RegisterUserRequest`.

```
RegisterUserRequest {  
    string Email;  
    string Password;  
}
```



Slika 3.21: Registracijska stran

### 3.6.3 Ustvarjanje, brisanje in odpiranje projektov

Po prijavi lahko uporabnik odpre stran s projekti. Na tej strani se prikažejo:

- seznam vseh njegovih projektov,
- gumbi za izbris posameznega projekta,
- gumb za dodajanje novega projekta in
- gumb za zahtevanje in izpust glasovnega pomočnika.

#### Ustvarjanje novega projekta

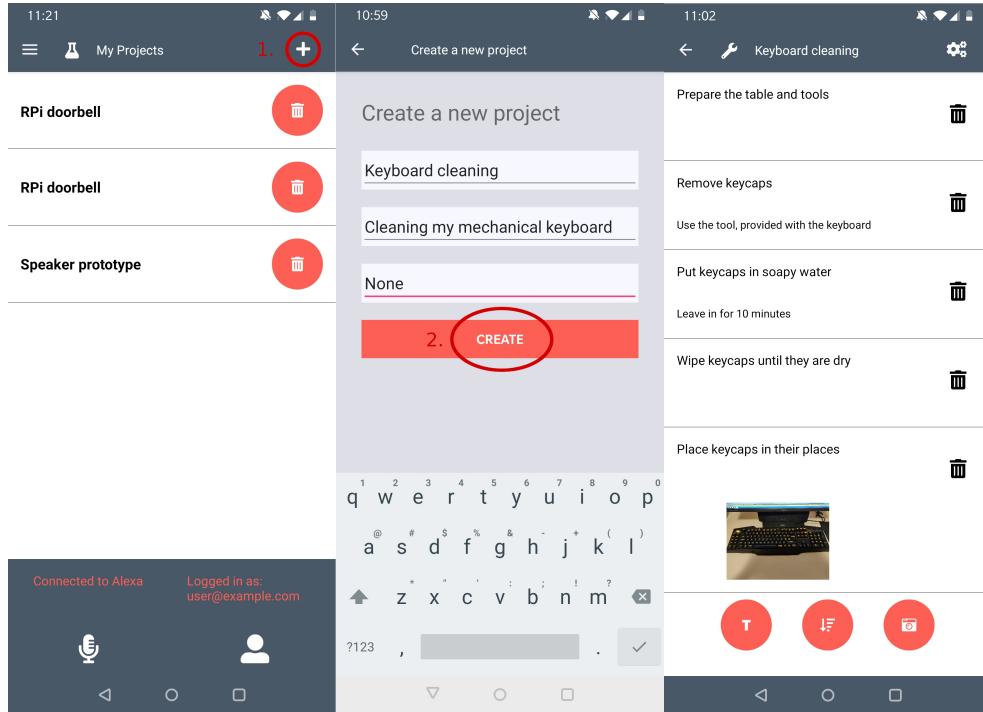
Za ustvarjanje novega projekta (slika 3.22), mora uporabnik odpreti stran za dodajanje projektov. Na tej strani mora vpisati naslov projekta (**Title**), kratek opis (**Description**) in našteti možne nevarnosti pri delu **Dangers**.

Te tri vrednosti se zapisa v objekt razreda `CreateProjectRequest`.

```
CreateProjectRequest {
    string Title;
    string Description;
    string Dangers;
}
```

Ta objekt nato storitev za komunikacijo s strežnikom pošlje preko metode POST na URL ”/projects/create”. Strežnik v odgovor vrne objekt razreda Project, ki ga je zahtevek ustvaril.

Nov projekt se nato prikaže v seznamu vseh uporabnikovih projektov na strani ”My Projects”, kjer ga lahko uporabnik odpre ali izbriše.



Slika 3.22: Postopek ustvarjanja novega projekta po korakih.

## Izbris projekta

Da uporabnik izbriše projekt, mora na strani ”My projects” ob željenem projektu pritisniti gumb z ikono smetnjaka. Pritisak na ta gumb sporoči sto-

ritvi za komunikacijo s strežnikom, naj pošlje avtoriziran DELETE zahtevek na URL ”/projects/{id}/delete”. Strežnik ob uspešnem izbrisu projekta odjemalcu vrne kopijo izbrisanega projekta.

### 3.6.4 Zahtevanje in izpust glasovnega pomočnika

Storitev za upravljanje z glasovnim pomočnikom (`VoiceAssistantService.cs`) skrbi za zahtevanje in izpust glasovnega pomočnika. Ta storitev se registrira ob začetku izvajanja aplikacije, nato pa počaka, da avtentikacijska storitev uspešno opravi prijavo uporabnika. Ob prijavi uporabnika preveri, ali ta uporablja glasovnega pomočnika. To doseže tako, da pošlje GET zahtevek na URL ”/users/current”. Strežnik vrne objekt `User`, ki priпадa prijavljenemu uporabniku. Storitev preveri vrednost boolean atributa `UsesVoiceAssistant`.

Če ima uporabnik vrednost polja `UsesVoiceAssistant` nastavljeno na `true`, lahko ima glasovnega pomočnika vezanega nase.

Če ima uporabnik vrednost polja `UsesVoiceAssistant` nastavljeno na `false`, mora uporabo glasovnega pomočnika zahtevati. To naredi tako, da pošlje avtoriziran PUT zahtevek na URL ”/users/requestva”.

Če noben drug prijavljen uporabnik nima polja `UsesVoiceAssistant` nastavljenega na `true`, se njegova zahteva odobri. V njegovem objektu `User` se nastavi vrednost `UsesVoiceAssistant` na `true`.

Če ima kateri od uporabnikov to polje nastavljeno na `true`, zahteva ne bo odobrena. V tem primeru mora uporabnik, ki zahteva pomočnika počakati, da uporabnik, ki pomočnika uporablja, pomočnika izpusti. Ko si pomočnika ne lasti nobeden drug uporabnik, ga lahko veže nase.

Izpust pomočnika dosežemo, da uporabnik, ki pomočnika trentuno uporablja, pošlje avtoriziran PUT zahtevek na URL ”/users/releaseva”.

V mobilni aplikaciji zahtevanje in izpust pomočnika nadziramo na ”Dashboard” strani, s pritiskom na ikono mikrofona (slika 3.23).



Slika 3.23: Gumb v obeh stanjih

### 3.6.5 Odpiranje in sinhronizacija projekta

Ko uporabnik iz seznama projektov izbere projekt, ki ga želi odpreti, storitev za komunikacijo s strežnikom pošlje na URL ”/projects/{id}/notes” zahtevek GET. Polje {id} je unikatni identifikator želenega projekta.

Storitev za projekte (razred `ProjectService.cs`) nato prejet odgovor, razreda `Project`, hrani. Vse korake, ki pripadajo temu projektu, se hrani v seznam tipa `ObservableCollection`. Ta tip seznama smo izbrali, saj je v ogrodju Xamarin nad njim najlažje opravljati zamenjevalne operacije. Seznam se nato prikažem na strani za upravljanje s projektom.

Storitev nato začne s periodično sinhronizacijo projekta. Storitev za projekte na URL ”/projects/{id}/nonotes” pošlje avtoriziran GET zahtevk. Strežnik vrne objekt razreda `Project` s praznim seznamom korakov. Vse, kar storitev za projekte pri sinhronizaciji gleda, je vrednost zastavic `NoteRequest`, `ImageRequest` in `AlexaNoteRequest`.

Na `true` nastavljena zastavica `NoteRequest` odpre obrazec za dodajanje novega besedilnega koraka v projekt.

Na `true` nastavljena zastavica `ImageRequest` odpre obrazec za dodajanje nove fotografije v projekt.

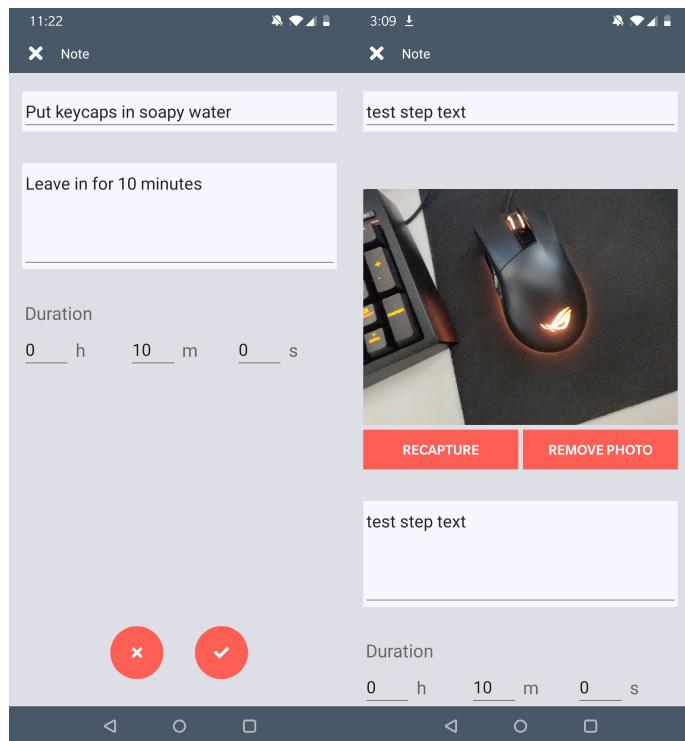
Na `true` nastavljena zastavica `AlexaNoteRequest` storitvi za projekte sporoči, naj pošlje GET zahtevk na URL ”/projects/{id}/notes”. Polje {id} mora biti unikatni identifikator odprtrega projekta. V odgovor prejme objekt razreda `Project`, vključno s seznamom vseh korakov, ki mu pripadajo. V prejetem seznamu je namreč tudi narekovan korak. Storitev za projekte

primerja, katerega od korakov še nima v svoji zbirki. Ko ta korak najde, ga doda v svoj ObservableCollection.

Ko se projekt sinhronizira, storitev za projekte pošlje na strežnik PUT zahtevo na URL ”/project/last/closerequests”. Ta zahteva nastavi vrednost zastavic NoteRequest, ImageRequest in AlexaNoteRequest na false.

### 3.6.6 Dodajanje in brisanje zapiskov

Ko ima uporabnik projekt odprt, lahko vanj dodaja korake, jih briše, ureja in jim spreminja vrstni red. Doda lahko tekstovni korak ali korak s fotografijo.



Slika 3.24: Stran za dodajanje in urejanje korakov

Pri dodajanju besedilnega koraka mora uporabnik v vnosnem obrazcu (slika 3.24) podati naslov koraka (Title), opis koraka (Text) ter trajanje v urah, minutah in sekundah. Ti podatki se nato hranijo v objektu razreda

**Note.** Ta objekt storitev za projekte vključi v objekt razreda `AddNoteRequest` in ga pošlje storitvi za komunikacijo s strežnikom.

```
AddNoteRequest {  
    Note note;  
}
```

Storitev za komunikacijo s strežnikom pošlje ta objekt na strežnik. Pošlje ga na URL ”/projects/{id}/addnote” preko metode POST. Polje `{id}` mora biti unikatni identifikator projekta, v katerega želimo korak vstaviti. Če je vstavljanje v projekt uspešno, strežnik odjemalcu pošlje v odgovor kopijo dodanega objekta `Note`.

Storitev za projekte nato ta korak doda svoji zbirki korakov.

Pri vstavljanju slikovnih korakov je postopek vstavljanja podoben. Ko uporabnik odpre stran za dodajanje slikovnega koraka, se zažene privzeta aplikacija za kamero. Ko uporabnik fotografijo posname, se ta prikaže na strani z vnosnim obrazcem (slika 3.24). Slika se na mobilni napravi hrani v mapi ”Pictures/{naslov projekta}”, v uporabnikovem domačem direktoriju. Uporabnik ima na vnosnem obrazcu možnost sliko ponovno posneti ali jo odstraniti.

Uporabnik mora v vnosnem obrazcu podati naslov koraka (`Title`), opis koraka (`Text`) ter trajanje v urah, minutah in sekundah.

Uporabnik mora nato pritisniti gumb za vstavljanje slikovnega koraka v projekt. Storitev za upravljanje s projektom fotografijo zakodira v znakovni niz po metodi Base64. Storitev za komunikacijo s strežnikom zakodirano fotografijo in novo ustvarjen objekt razreda `Note` vključi v objekt tipa `UploadImageRequest`.

```
UploadImageRequest {  
    Note note;  
    string ImageString;  
}
```

Novo nastali zahtevek `UploadImageRequest` se nato avtorizira z JWT avtorizacijskim žetonom in pošlje na URL ”/projects/{id}/addimage”. Polje `{id}` mora biti unikatni identifikator projekta, v katerega želimo korak vstaviti. Če je vstavljanje v projekt uspešno, strežnik odjemalcu pošlje v odgovor kopijo pravkar vstavljenega objekta `Note`.

## Uporaba kamere v Xamarin

Za uporabo kamere v ogrodju Xamarin smo uporabili vtičnik `Xam.Plugin.Media` [13]. Vtičnik je licensiran pod MIT licenco.

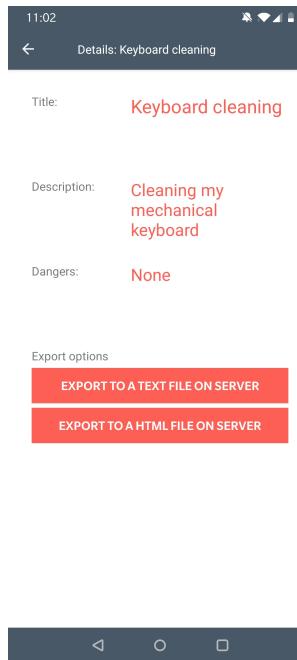
Zanj smo se odločili, saj podpira zajem fotografij s privzeto aplikacijo za zajemanje fotografij. Izbiramo lahko tudi kvaliteto zajete fotografije.

Pri zajemu slike najprej preverimo, ali imamo zadostne pravice. Imeti moramo pravice za dostop do shrambe, dostop do fotografij in dostop do kamere.

Fotografijo nato zajamememo z metodo `TakePictureAsync`. Kreirano fotografijo nato preberemo in znotraj aplikacije hranimo v objektu razreda `ImageSource`. Fotografijo v tej obliki lako prikažemo v XAML z grafičnim elementom `Image`. Lahko jo tudi zakodiramo v znakovni niz in pošljemo na strežnik.

### 3.6.7 Izvoz projekta

Uporabnik lahko projekt izvozi v človeško berljivo obliko na strežnik v dva različna formata: besedilno datoteko (.txt) in HTML dokument.



Slika 3.25: Stran za izvoz projektov

Če želi uporabnik izvoziti projekt kot besedilno datoteko, storitev za komunikacijo s strežnikom pošlje POST zahtevek na URL ”/project/{id}/export/text”. Če želi uporabnik izvoziti projekt kot HTML dokument, storitev za komunikacijo s strežnikom pošlje POST zahtevek na URL ”/project/{id}/export/html”.

## 3.7 Testiranje sistema OpenReport

## 3.8 Ovrednotenje funkcionalnosti

Primerjali smo pisanje opisa tehnološkega postopka na list papirja, s pisarniškim programom in s sistemom OpenReport. Sistem OpenReport smo testirali z uporabo Amazon Alexe in brez uporabe Amazon Alexe.

Mobilna aplikacija in strežnik sta izpolnila pričakovanja. Možnost pisanja tekstovnih korakov je smiselna za manjše opombe. Uporaba slikovnih korakov je smiselna za večje oporne točke pri delu.

Možnost izvoza projekta omogoča tudi prenos podatkov v druge računalniške programe.

Hitrost uporabe sistema se z uporabo glasovnega pomočnika ni izboljšala.

Pod pričakovanjimi se je izkazalo narekovanje besedila glasovnemu pomočniku. Razpoznavanje glasu pri „Skill-ih“ za pomočnika Amazon Alexa namreč deluje najbolje, ko ima stavek veliko stopnjo ujemanja z definirano frazo.

- Alexa je ukaz ”Take a picture“ pravilno razpoznala v 18 od 20 testnih izgovorjav.
- Ukaz ”Create a text note“ je pravilno razpoznala v 17 od 20 testnih izgovorjav.
- Ukaz ”note this is a voice note“ je pravilno razpoznala v 3 od 20 testnih izgovorjav.
- Ukaza ”note backplate removal takes about three minutes“ ni pravilno razpoznala v nobeni od 20 testnih izgovorjav.

Dodajanje slikovnih in besedilnih korakov, s pomočjo glasovnega ukaza, se je izkazalo za delno uspešno. Uporabno je bilo predvsem, ko mobilnega telefona med delom nismo imeli na dosegu roke. Kljub temu je problem prikazovalo dejstvo, da je izgovorjava invokacijske fraze, fraz namer in čakanje na obdelavo zahtev, trajalo v povprečju 18 sekund.

Odklepanje mobilnega telefona, ki ga imamo v dosegu rok in ga lahko dosežemo brez presedanja ter odpiranje forme za dodajanje korakov je trajalo v povprečju 4 sekunde. Odklepanje mobilnega telefona, ki je oddaljen tri metre ter odpiranje forme za dodajanje korakov je trajalo povprečju 8 sekund.



# Poglavlje 4

## Zaključek

Sistem, razvit v sklopu diplomske naloge zajema funkcionalosti, ki uporabniku omogočajo sestavljanje delavnih dnevnikov. Sestavljajo ga strežnik, mobilna aplikacija in glasovni pomočnik Amazon Alexa. Uporabnik lahko v delavnike dodaja preproste tekstovne korake. Z uporabo mobilne aplikacije lahko zajame fotografije in jih v delavnški dnevnik vstavi kot slikovno gradivo.

Ta sistem je služil kot primer sistema, ki ga želimo izboljšati z uporabo glasovnega pomočnika. Naš namen je bil poenostaviti delo s sistemom, z uvedbo glasovnega pomočnika Amazon Alexe. Implementirali smo odpiranje kamere in obrazca za dodajanje korakov z glasovnim pomočnikom. Kljub uspešni implementaciji, je bil čas, potreben za izgovorjavo in obdelavo glasovnih ukazov, predolg, da bi poskus izboljšave bil popolnoma uspešen. Za neuspešno se je izkazalo glasovno narekovanje vsebine koraka delavnškega dnevnika. Z Amazon Alexo ni bilo mogoče razpoznati daljših, variirajočih glasovnih ukazov.

V možne izboljšave sistema smo na prvo mesto uvrstili preizkus drugih glasovnih pomočnikov, npr. Google Assistant [8]. Večjo učinkovitost bi lahko dosegli z pomočnikom, ki bi lahko razpoznał ukaze z eno samo frazo in ne kombinacijo dveh, kot Amazon Alexa. Druga smiselna izboljšava je izvoz delavnih dnevnikov v več različnih standardnih formatov, kot so npr. XSL.

Mobilna aplikacija in strežniški program sta zadovoljila pričakovanja in lahko že v trenutnem stanju služita kot referenca za primerjavo učinkovitosti Amazon Alexe in drugih glasovnih pomočnikov.

Kljub nekaterim težavam z uporabo glasovnega pomočnika, je očitno, da je tehnologija glasovnih pomočnikov zelo primeren način za nadzor računalniških sistemov na daljavo. Poleg tega ponuja veliko možnosti za slepe, slabovidne in gibalno omejene posameznike, ki imajo težave z interakcijo z računalnikom.





# Literatura

- [1] *.NET ogrodje*. Dosegljivo: <https://dotnet.microsoft.com/>. [Dostopano: 18. 8. 2020].
- [2] *Amazon Alexa*. Dosegljivo: <https://www.amazon.com/b?ie=UTF8&node=17934671011>. [Dostopano: 12. 8. 2020].
- [3] *Arhitektura Alexa „Skill-a“*. Dosegljivo: <https://blogs.sap.com/2016/10/10/b1-assistant-retire-keyboard-sap-business-one-alexa/>. [Dostopano: 28. 8. 2020].
- [4] Jonas Austerjost, Marc Porr, Noah Riedel, Dominik Geier, Thomas Becker, Thomas Scheper, Daniel Marquard, Patrick Lindner in Sascha Beutel. „Introducing a virtual assistant to the lab: A voice user interface for the intuitive control of laboratory instruments“. V: *SLAS TECHNOLOGY: Translating Life Sciences Innovation* 23.5 (2018), str. 476–482.
- [5] *Delavníški dnevnik kot izobraževalni pripomoček*. Dosegljivo: [https://www.gzs.si/Portals/203/Vsebine/novice-priponke/DNEVNIK20za20vajenistvo\\_2019\\_navodila20z20zgledom.pdf](https://www.gzs.si/Portals/203/Vsebine/novice-priponke/DNEVNIK20za20vajenistvo_2019_navodila20z20zgledom.pdf). [Dostopano: 18. 8. 2020].
- [6] *Funkcionalnosti AWS Lambda*. Dosegljivo: <https://aws.amazon.com/lambda/features/>. [Dostopano: 12. 8. 2020].
- [7] *Funkcionalnosti AWS SQS*. Dosegljivo: <https://aws.amazon.com/sqs/features/>. [Dostopano: 12. 8. 2020].

- [8] *Google Assistant*. Dosegljivo: <https://assistant.google.com/>. [Dostopano: 18. 8. 2020].
- [9] *LibreOffice Writer*. Dosegljivo: <https://www.libreoffice.org/discover/writer/>. [Dostopano: 12. 8. 2020].
- [10] *Navodila za pisanje delavnškega dnevnika ŠCLJ*. Dosegljivo: [https://www.lesarska.sclj.si/images/Pravilniki/NOZ\\_PRA.doc](https://www.lesarska.sclj.si/images/Pravilniki/NOZ_PRA.doc). [Dostopano: 28. 8. 2020].
- [11] *Shema MVVM pristopa*. Dosegljivo: <https://docs.devexpress.com/WPF/images/winforms-mvvm-common-mvvm-scheme118014.png>. [Dostopano: 18. 8. 2020].
- [12] *Uvod v Alexa "Skill-e"*. Dosegljivo: <https://dotnet.microsoft.com/apps/xamarin>. [Dostopano: 12. 8. 2020].
- [13] *Xam.Plugin.Media*. Dosegljivo: <https://github.com/jamesmontemagno/MediaPlugin>. [Dostopano: 12. 8. 2020].
- [14] *Xamarin*. Dosegljivo: <https://dotnet.microsoft.com/apps/xamarin>. [Dostopano: 12. 8. 2020].
- [15] *Žetoni JWT kot sredstvo za avtorizacijo*. Dosegljivo: <https://jwt.io/introduction/>. [Dostopano: 28. 8. 2020].